

GEORGIA SOFTWARES

GSWBrowse for Windows RF Devices

Industrial Browser for RF Devices

User Manual

THIS PAGE INTENTIONALLY LEFT BLANK

GEORGIA SOFTWORKS

GSWBrowse Web Client

Copyright © 1997-2011, Georgia SoftWorks, All Rights Reserved
Public Square
17 Hwy 9 South • PO Box 729
Dawsonville Georgia 30534
Telephone 706.265.1018 • Fax 706.265.1020
[Visit GSW Web Site](#)

Copyright © Georgia SoftWorks, 1997-2011 All Rights Reserved.

User's Manual, Version 1.07.0003, Mar 03, 2008,

Microsoft, Windows, Windows CE, Windows XP, Windows 2000, Windows 2003, VISTA, Windows NT, Windows 98, Windows 95 are trademarks of Microsoft Corporation. SAP, SAPConsole are trademarks of SAP AG. LXE, is a trademark of LXE

THIS PROGRAM IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

LICENSOR MAKES NO WARRANTIES OR REPRESENTATIONS, EXPRESSED OR IMPLIED, ORAL OR WRITTEN, REGARDING THE PROGRAM OR DOCUMENTATION AND HEREBY EXPRESSLY DISCLAIMS ALL OTHER EXPRESSED OR IMPLIED WARRANTIES, INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. LICENSOR DOES NOT WARRANT THE PROGRAM WILL MEET YOUR REQUIREMENTS OR THAT IT'S OPERATION WILL BE UNINTERRUPTED OR ERROR FREE.

IN NO EVENT WILL GEORGIA SOFTWORKS BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAMS.

COPYING:

WHILE YOU ARE PERMITTED TO MAKE BACKUP COPIES OF THE SOFTWARE FOR YOUR OWN USE AND PROTECTION, YOU ARE NOT PERMITTED TO MAKE COPIES FOR THE USE OF ANYONE ELSE.

LICENSE:

YOU ARE LICENSED TO RUN THIS SOFTWARE ON A SINGLE WINDOWS CE SYSTEM. THE GEORGIA SOFTWORKS GSWBROWSE SOFTWARE MAY BE INSTALLED ON A SINGLE WINDOWS CE SYSTEM.

This Page Left Intentionally Blank

Table of Contents

TABLE OF TABLES	IX
TABLE OF EXAMPLES.....	IX
TYPOGRAPHIC CONVENTIONS.....	X
FEATURES AT A GLANCE	XI
OVERVIEW.....	1
INSTALLATION.....	3
REGISTRATION/LICENSEING.....	4
INDIVIDUAL (LOCAL LICENSE) GSWBROWSE REGISTRATION.....	5
NETWORK (MULTIPLE) REGISTRATION.....	8
<i>Network Registration Setup</i>	11
GSWBrowse.txt	11
License Server	11
License Server Page.....	12
License Server Port	12
License File URL.....	12
EVALUATION LICENSE.....	13
GSWBROWSE LAYOUT.....	15
GSWBROWSE – MENU BAR – FILE - SECURITY LEVEL	16
<i>Security Mode Dialog</i>	17
<i>Administrator Security Mode</i>	17
<i>User Security Mode</i>	18
<i>Lock Down</i>	18
GSWBROWSE – MENU BAR – EDIT	20
<i>Printer Configuration Overview</i>	20
<i>Local (Serial/Bluetooth/IrDA)</i>	22
<i>Network Printer</i>	23
GSWBROWSE – MENU BAR – VIEW	24
<i>Licensing</i>	24
<i>Set As Home Page</i>	24
<i>Reset Home Page</i>	24
<i>Enabled</i>	24
<i>Multiple Browsers</i>	25
<i>Browsers</i>	26
<i>Bars</i>	27
<i>UI Control – User Interface Control</i>	29
Context Menus Overview.....	30
Context Menus	30
Default Context Menu.....	30
Image Context Menu.....	30
Control Context Menu.....	31
Table Context Menu.....	31
Text Context Menu	31
Anchor Context Menu.....	31
Unknown Context Menu	31
No Text Selection.....	31
No 3D Border.....	32
No Scrollbars.....	32
<i>Dld Control – Download Control</i>	33
Images	33

Videos	33
Sounds	33
No Script Execution	33
No Java Applets	34
No Execution of ActiveX controls	34
No Client Pull.....	34
No Frames download	34
Force Offline	34
DEVELOPERS GUIDE	35
OVERVIEW	35
GSW Action Codes	35
window.external extensions	36
In summary	36
GSW HTML META TAGS OVERVIEW	37
Meta Tag Syntax	38
NAVIGATIONAL SYNTAX	38
Anchor Tag	39
HREF Property	39
NAVIGATE Method.....	41
WINDOW EXTERNAL SYNTAX	42
SEND MENU COMMANDS OVERVIEW	43
DEVELOPERS QUICK REFERENCE	44
Hooking the Keyboard	44
Battery.....	44
Resume Device Operation after Suspension	44
Timer-Based Actions.....	44
Browser Navigation Error Handling	44
Exit from Browser	44
Controlling Input Panel (soft keyboard).....	44
Date and Time.....	45
Font Size	45
Cursor Position.....	45
Warm Boot and Soft Boot.....	45
RF Signal Strength.....	45
Scanner Operation	45
GSWBrowse Application Control	45
Printing (BlueTooth, IRDA, Network, Serial)	45
GSWBrowse about Information	46
Persistent Storage	46
Device Information	46
Network Adapter Name.....	46
Vibrating the Device (LXE Only).....	46
Controlling the Taskbar.....	46
Vendor API Version (LXE Only).....	46
LXE SCANNER APIs	47
GSWBrowse Interface Helper Object	48
DEVELOPERS DETAILED REFERENCE	49
Scope / Syntax	49
Developer Attention Boxes.....	49
HOOKING THE KEYBOARD DETAIL	51

gswbrowse_ONKEY.....	51
gswbrowse_ONALLKEYS.....	53
gswbrowse_HOMEKEY.....	55
BATTERY INFORMATION.....	56
gswbrowse_BATTERYNAVIGATE.....	56
gswbrowse_BATTERYTIMERINTERVAL.....	59
window.external.device.batterystatus	61
window.external.device.battery.status	63
window.external.device.battery.statustext.....	64
window.external.device.battery.level	65
RF SIGNAL STRENGTH.....	66
gswbrowse_SIGNALNAVIGATE.....	66
window.external.device.rf.strength	70
window.external.device.rf.status	72
RESUME AFTER SUSPENSION	74
gswbrowse_POWERON.....	74
SCANNER OPERATION.....	76
gswbrowse_SCANNERNAVIGATE.....	76
TIMER-BASED ACTIONS	78
gswbrowse_TIMERNAVIGATE	78
gswbrowse_TIMERINTERVAL	80
BROWSER NAVIGATION ERRORS	82
gswbrowse_ERRORNAVIGATE	82
EXITING BROWSER	85
gswbrowse_COMMAND.....	85
gswbrowse_APPLICATION.....	86
Soft Input Panel (SIP).....	87
gswbrowse_SIP	87
gswbrowse_SIPUP.....	89
gswbrowse_MOVESIP.....	91
TIME AND DATE.....	93
gswbrowse_SETTIME.....	93
gswbrowse_SETDATE	94
FONT SIZE CONTROL	96
gswbrowse_FONTSIZE.....	96
CURSOR POSITIONING.....	98
gswbrowse_CURSORPOS	98
REBOOTING DEVICE – WARM/SOFT.....	100
gswbrowse_REBOOT.....	100
SEND MENU COMMANDS.....	102
window.external.app.sendmenucommand.....	102
SEND MENU SYMBOLIC NAMES AND LITERALS.....	105
PRINTING	108
gswbrowse_ZEBRALABEL_PRINT.....	108
gswbrowse_ZEBRALABEL_COMPLETE.....	110
window.external.app.session(idx).printer.PrintString	111
BROWSER AND SESSION DATA.....	112
window.external.app.Session	112
window.external.ThisSessionIndex	114
window.external.app.ActiveSessionIndex	115
window.external.app.session(idx).host	117
window.external.app.session(idx).port	119
window.external.app.session(idx).type.....	121

window.external.app.session(idx).path	123
window.external.app.session(idx).homepage	125
window.external.app.version	127
window.external.app.buildid	128
window.external.app.name	129
window.external.app.copyright	130
PERSISTENT STORAGE	131
window.external.app.usertext	131
window.external.app.activesession	132
DEVICE INFORMATION	134
window.external.device.deviceid	134
window.external.device.ipaddress	135
window.external.device.macaddress	136
window.external.device.platformid	137
window.external.device.presetid	138
window.external.device.serialnumber	139
window.external.device.uuid	140
NETWORK ADAPTER	142
window.external.device.networkadaptername	142
VIBRATING THE DEVICE	143
window.external.device.vibrate	143
TASKBAR CONTROL	144
window.external.device.settaskbarenable	144
window.external.device.settaskbarvisible	145
window.external.device.setstartbuttonvisible	146
window.external.device.setsipbuttonvisible	148
VENDOR API VERSION	149
window.external.device.apiversion	149
LXE DEVICE API'S	150
window.external.device.scanner.lxescanner.LXEHASINTSCANNER	152
window.external.device.scanner.lxescanner.LXEHASINTRFID	153
window.external.device.scanner.lxescanner.LXESCANNERDISABLE	154
window.external.device.scanner.lxescanner.LXESCANNERENABLE	155
window.external.device.scanner.lxescanner.LXESCANNERINPUTDISABLE	157
window.external.device.scanner.lxescanner.LXESCANNERINPUTENABLE	158
window.external.device.scanner.lxescanner.LXESCANNERRESET	159
window.external.device.scanner.lxescanner.LXESCANNERFORCEPOWER	161
window.external.device.scanner.lxescanner.LXESCANNERFORCESCAN	163
window.external.device.scanner.lxescanner.LXESCANNERSTART	164
window.external.device.scanner.lxescanner.LXESCANNERSTOP	166
window.external.device.scanner.lxescanner.LXESCANNERGETSTATUS	167
window.external.device.scanner.lxescanner.LXESCANNERGETDATA	169
window.external.device.scanner.lxescanner.LXESCANNERPUTDATA	171
window.external.device.scanner.lxescanner.LXESCANNERPOWERON	172
window.external.device.scanner.lxescanner.LXESCANNERPOWEROFF	173
window.external.device.scanner.lxescanner.LXESCANNERKEYSOFF	174
window.external.device.scanner.lxescanner.LXESCANNERKEYSON	175
window.external.device.scanner.lxescanner.LXESCANNERKEYSTATUS	176
window.external.device.scanner.lxescanner.LXESCANNERATTACHPORT	177
window.external.device.scanner.lxescanner.LXESCANNERSETKEY	179
window.external.device.scanner.lxescanner.LXESCANNERSETSERIAL	181
window.external.device.scanner.lxescanner.LXESCANNERSETPOWER	183

<i>window.external.device.scanner.lxescanner.LXESCANNERSTRIPLEAD</i>	185
<i>window.external.device.scanner.lxescanner.LXESCANNERSTRIPTRAIL</i>	186
<i>window.external.device.scanner.lxescanner.LXESCANNERCTRLCODEOFF</i>	187
<i>window.external.device.scanner.lxescanner.LXESCANNERCTRLCODEON</i>	188
<i>window.external.device.scanner.lxescanner.LXESCANNERPREFIX</i>	189
<i>window.external.device.scanner.lxescanner.LXESCANNERSUFFIX</i>	190
<i>window.external.device.scanner.lxescanner.LXESCANNEROPTION</i>	191
<i>window.external.device.scanner.lxescanner.LXESCANNERISOPTIONENABLED</i>	193
<i>window.external.device.scanner.lxescanner.LXESCANNERGETPARAM</i>	195
<i>window.external.device.scanner.lxescanner.LXESCANNERSETPARAM</i>	197
<i>window.external.device.scanner.lxescanner.LXESCANNERSYMSTRIPLEAD</i>	199
<i>window.external.device.scanner.lxescanner.LXESCANNERSYMSTRIPTRAIL</i>	201
<i>window.external.device.scanner.lxescanner.LXESCANNERSYMPREFIX</i>	203
<i>window.external.device.scanner.lxescanner.LXESCANNERSYMSUFFIX</i>	205
<i>window.external.device.scanner.lxescanner.LXESCANNERSYMENABLE</i>	207
<i>window.external.device.scanner.lxescanner.LXESCANNERSYMISENABLED</i>	209
<i>window.external.device.scanner.lxescanner.LXESCANNERSYMGETCONFIG</i>	211
<i>window.external.device.scanner.lxescanner.LXESCANNERSYMSETCONFIG</i>	213
<i>window.external.device.scanner.lxescanner.LXESCANNERSYMREMOVE</i>	215
GSW HELPER OBJECT	216
<i>window.external.interface.CREATEPARAMETERBLOCK</i>	220
SELECTED SAMPLES FROM CD	224
Batt_NAV.htm.....	224
DATETIME.htm.....	225
EXTERNAL.HTM.....	226
LXE Helper Object.HTM.....	228
DEVELOPER OBJECTS AND CONSTANTS REFERENCE	230
GSW PROGRAMMING OBJECTS – INTERFACE DESCRIPTION LANGUAGE.....	230
GSWBROWSE:Idispatch.....	230
IExternalApp:Idispatch.....	230
IExternaldevice:Idispatch.....	231
IExternalSession:Idispatch.....	232
IExternal:PrintString.....	232
IExternalBattery:Idispatch.....	232
IExternalRF:Idispatch.....	232
IExternalScanner:Idispatch.....	232
IExternaldeviceLXEScanner:Idispatch.....	233
IExternalInterface:Idispatch.....	234
IExternalJSParameters:Idispatch.....	234
GSW PROGRAMMING STRUCTURES/VALUES/CONSTANTS.....	235
LXE DEVICE CONSTANTS.....	237
<i>Bitmap Values for Scan Code Types</i>	238
<i>Symbologies</i>	239
<i>Parms for LXEScannerOption</i>	240
<i>Flag Parms for LXEScanner Get/Set Param</i>	240
VIRTUAL KEY CODES.....	241
IMPORTANT FILES	244
GSWBROWSE.TXT.....	244
LICENSE.ASP.....	245
WRITELOG.ASP.....	246

LICENSE.XML	247
GSWBROWSE SUBSCRIPTION.....	248
HOW TO UPDATE THE SOFTWARE	248
HOW TO RENEW THE GSWBROWSE SOFTWARE.....	248
SYSTEM SIGNATURE - IMPORTANT PLEASE READ.....	249
TECHNICAL SUPPORT.....	250

TABLE OF FIGURES

Figure 1: Licensed and Evaluation Diagram.....	4
Figure 2: GSWBrowse Registration Techniques.....	5
Figure 3: Menu Bar View -> Licensing	5
Figure 4: Registration Dialog.....	6
Figure 5: Registration Dialog - Customer Name and Save	6
Figure 6: Network Registration – High Level Overview - Multiple GSWBrowse's	8
Figure 7: Network Registration Setup File Diagram	10
Figure 8: Licensed and Evaluation Flow Diagram	13
Figure 9: Evaluation Mode Pop-up.....	14
Figure 10: GSWBrowse Layout	15
Figure 11: GSWBrowse - File->Menu->Security Mode	16
Figure 12: Security Mode Dialog	16
Figure 13: Administrator Security Mode.....	17
Figure 14: Switch to User Mode.....	18
Figure 15: Menu bar - Edit (Printer Configuration)	20
Figure 16: Printer Configuration - Enable Client Side Printing	21
Figure 17: Network and Serial/Bluetooth/IrDA options enabled.....	21
Figure 18: Enable Client Side Printing.....	22
Figure 19: Local Printer port selection and configuration properties	22
Figure 20: Printing - Network Printer Configuration	23
Figure 21: Menu Bar - View	24
Figure 22: Multiple Browsers Configuration	25
Figure 23: Multiple Web Browsers Enabled/Disabled.....	25
Figure 24: Select Browser Menu Item.....	26
Figure 25: Browsers Tab	26
Figure 26: Diagram of Browser Bars	27
Figure 27: Maximize Content Area - Hide Bars.....	28
Figure 28: Maximize Content Area	28
Figure 29: UI Control Menu.....	29
Figure 30: UI Control - Text Context Menu	30
Figure 31: Dld (Download Control)	33
Figure 32: GSW Commands Action Code Definition Diagram	36
Figure 33: Meta Tag Processing Overview	37
Figure 34: Meta Tag Syntax Example	38
Figure 35: Navigational Syntax Properties, Methods and HTML.....	38
Figure 36: Georgia SoftWorks Navigation Syntax Parts.....	39
Figure 37: Meta Tag Syntax Compared with Navigational Syntax.....	39
Figure 38: Navigation Syntax - HREF Property	40
Figure 39: Navigation Syntax - NAVIGATE Method.....	41
Figure 40: GSWBrowse Programmable External Objects	42
Figure 41: Syntax / Scope Diagram	49
Figure 42: Passing Battery Status to Your JavaScript Function.....	57
Figure 43: Battery Navigate - Passing Form Data as NAME/VALUE pairs.....	58
Figure 44: Passing Signal Status Information to Your JavaScript Function.....	67
Figure 45: Signal Navigate - Passing Form Data as Name/Value Pairs.	68
Figure 46: Timer Navigate - Passing the time to your JavaScript function	79
Figure 47: Timer Navigate - Passing the time to you Form (URL).	79
Figure 48: Error Navigate - Passing the OS Error Number and Error Message to your JavaScript function.....	83
Figure 49: Error Navigate - Passing the Error Number and Message to a URL	83

Figure 50: SIPUp Coordinate Flexible Syntax Specifications	92
Figure 51: CURSORPOS Coordinate Flexible Syntax Specification	99
Figure 52: Host part of URL	117
Figure 53: Type part of URL	121
Figure 54: Path part of URL	123
Figure 55: Session Object	132
Figure 56: GSWBrowse Helper Object	150
Figure 57: LXE Scanner Object / Methods	151
Figure 58: GSWBrowse Helper Object - Overview	217
Figure 59: GSW Helper Object - Overview - Create Parameter Block	218
Figure 60: GSW Helper Object - Overview - Load Values	218
Figure 61: GSWBrowse Helper Object - Overview - Retrieve Data	219

Table of Tables

Table 1: Device Setup Programs	3
Table 2: Send Menu Command – File	105
Table 3: Send Menu Command – Edit	105
Table 4: Send Menu Command - View	105
Table 5: Send Menu Command - View Bars	105
Table 6: Send Menu Command - View Browsers	106
Table 7: Send Menu Command - View UI Control	106
Table 8: Send Menu Command - View DId Control	106
Table 9: Send Menu Command - GO	107
Table 10: Send Menu Command - Help	107

Table of Examples

Example: GSWBROWSE_ONKEY	52
Example: GSWBROWSE_ONALLKEYS	54
Example: GSWBROWSE_BATTERYTIMERINTERVAL	60
Example: BATTERY STATUS	62
Example: GSWBROWSE_SIGNALNAVIGATE	69
Example: WINDOW.EXTERNAL.DEVICE.RF.STRENGTH	71
Example: WINDOW.EXTERNAL.DEVICE.RF.STATUS	73
Example: GSWBROWSE_POWERON	75
Example: GSWBROWSE_SCANNERNAVIGATE	77
Example: GSWBROWSE_TIMERNAVIGATE	81
Example: GSWBROWSE_TIMERINTERVAL	81
Example: GSWBROWSE_ERRORNAVIGATE	84
Example: GSWBROWSE_COMMAND	85
Example: GSWBROWSE_APPLICATION	86
Example: GSWBROWSE_SIP	88
Example: GSWBROWSE_SIPUP	92
Example: GSWBROWSE_SETDATE	95
Example: GSWBROWSE_SETTIME	95
Example: GSWBROWSE_FONTSIZE	97
Example: GSWBROWSE_CURSORPOS	99
Example: GSWBROWSE_REBOOT	101
Example: SENDMENUCOMMAND 1	103
Example: SENDMENUCOMMAND 2	104
Example: GSWBROWSE_ZEBRALABEL_PRINT	109
Example: WINDOW.EXTERNAL.APP.SESSION	113
Example: WINDOW.EXTERNAL.APP.ACTIVESESSIONINDEX	116
Example: WINDOW.EXTERNAL.APP.SESSION(IDX).HOST	118
Example: WINDOW.EXTERNAL.APP.SESSION(INDEX).PORT	120
Example: WINDOW.EXTERNAL.APP.SESSION(IDX).TYPE	122
Example: WINDOW.EXTERNAL.APP.SESSION(IDX).PATH	124
Example: WINDOW.EXTERNAL.APP.SESSION(IDX).HOMEPAGE	126



Typographic Conventions

<i>Italics:</i>	are used to emphasize certain words, especially new terms or phrases when they are introduced.
Initial Caps Bold:	Words that appear in initial caps boldface represent menu options, buttons, icons or any object that you may click.
Courier:	This font represents anything you must type. Courier is used for examples.
"<enter>"	This represents the enter key.

Terms/Abbreviations

UTS	GSW Universal Terminal Server
Windows	Refers to Microsoft Windows Operating Systems 98/ME/NT 4.0/XP/VISTA/2000/2003 and CE unless otherwise noted.



Features at a Glance

Georgia SoftWorks GSWBrowse

- Support for Windows CE 5.0, CE 4.2 .NET
- Powerful Browser Customized for RF Devices and Barcode Scanners
- Device status and control features (e.g.: Battery Status, Signal Strength, etc.)
- Multiple Host Sessions, Provide Up To 5 Browser Instances to the web server enhancing speed. Reminiscent of IE7
- Printer support on a per browser basis. Bluetooth, IrDA, Serial, Network
- Works “Out of the Box”
- Easy to Install and Use
- Multiple Registration Methods to suite YOUR needs
- Accelerator Keys
- End User Lock Out
- Administrator Friendly



GSWBROWSE

INDUSTRIAL WEB BROWSER
FOR RF DEVICES

JANUARY 15, 2011



Overview

GSWBrowse is a Windows CE compatible nonsense free industrial grade browser customized for RF Terminals, Barcode Scanners and other Data Collection Devices

Thank you for purchasing the Georgia SoftWorks GSWBrowse for RF Devices and Barcode Scanners. You will find GSWBrowse provides a reliable, secure and consistent solution for browser based business/industrial applications. GSWBrowse is content compatible with Windows CE Versions 4.2, 5.0 and higher versions of Internet Explorer. The user interface is intuitive with a familiar and easy to understand look and feel.

Even though GSWBrowse operates “out of the box”, it is also configurable and programmable providing the flexibility to optimize the browser to your specific environment. Wide compatibility of GSWBrowse enables operation on heterogeneous devices able to connect to most web servers. Developers have abundant capabilities enabling rapid development of powerful applications with advanced yet easy to use features. Up to **five** browsers are available via quick swap tabs providing streamlined access to multiple pages.

GSWBrowse has three primary groups of users, each with their own important yet different roles.

- System Administrators
- Developers
- End Users (The Ultimate User)

Even though they have different tasks, the developer can programmatically perform many of the same configuration operations as the system administrator performs manually. In this respect they have overlapping capabilities. The efforts performed by the developer and system administrator come to fruition though the work of the end user.

System Administrators will find GSWBrowse is straightforward to install, register, configure lockdown and deploy. Administrators are provided with power over many situational specific factors. Each GSWBrowse browser can be independently configured.

Administrators can configure the browser such that the user is **completely locked out** of the operating system and any other user interface, ensuring access only to the content intended by the web application. Limiting access to only necessary items makes the job more efficient and less complicated for everyone including the user.

Additionally GSWBrowse can maximize the viewable content area by configuring the layout of the browser by enabling/disabling user interface components such as captions and tool bars, status, address etc.

See more about the layout of GSWBrowse and configuration of GSWBrowse on [page 15](#).

System administrators can depend on the reliable and consistent operation of GSWBrowse throughout daily activities as well as across reboots.



Developers writing the content for the application can exploit browser enhanced features to programmatically control the device as well as obtain and present status information. Providing users with battery status, RF Signal strength etc. becomes simple using standard web page development and scripting tools. With this data users can complete their job with awareness and effectiveness.

GSWBrowse provides these capabilities through clear and quick to learn JavaScript and Meta Tag interfaces.

GSWBrowse provides programmable objects allowing access to Windows CE functionality and GSWBrowse itself. Programmable objects are also used to access [Generic](#) and [Vendor specific](#) features of the device. Programmable objects and Meta Tags are utilized in order to provide a rich set of features for the developer. This goes far beyond the capabilities of the standard browser which is typically unaware of the device and its capabilities. See more in the Developer Guide section on [page 35](#).

With a typical browser, navigation controls are built in and available to the end user. By controlling access to these navigation objects with GSWBrowse, the developer can guide end user navigation and work flow. It is easier to create applications when the developer controls all navigation elements and is able to specify exactly the navigation options available to the user. This simplifies the need of precisely monitoring work flow as well as eliminating a multitude of error cases.

End Users are able to focus on their work with a browser that minimizes or eliminates distracting controls and objects. The system administrator and the developer can set up the User Interface so the end users can perform their particular job in the most efficient way possible. The steep learning curve avoided by browser customization proves profitable. There is no need to worry about accidentally misconfiguring the device or accidentally navigating to an incorrect web site.



Installation

GSWBrowse is straightforward to install on your device.

1. Determine the location and name of the GSWBrowse setup program for your device. Please see Table 1 for the setup programs available.
2. Copy the appropriate setup executable to the computer which *established partnership* with your device.
3. Turn on your device
4. Start the Active Sync on the device and connect.
5. Run the self-extracting executable on the computer selected in Step 2
6. Confirm the prompts asking to continue with the installation.
7. You should see the setup program launched on your device.
8. Press the OK button on the device to complete the installation.

Device Manufacturer	Architecture	Operating System	GSW Scanner Extensions	Setup Program
LXE	ARMv4i	WIN CE 4.2	Yes	BrLXE.exe
LXE	ARMv4i	WIN CE 5.0	Yes	BrLXE.exe
Psion-Teklogix	ARMv4i	WIN CE 4.2	Yes	BrPT420.exe
Psion-Teklogix	ARMv4i	WIN CE 5.0	Yes	BrPT500.exe
Intermec CK30	ARMv4i	WIN CE 4.2	Yes	BrCK30.exe
Standard	ARMv4i	WIN CE 4.2	No	BrARMv4i.exe
Standard	ARMv4i	WIN CE 5.0	No	BrARMv4i.exe
Standard	ARMv4	WIN CE 4.2	No	BrARMv4.exe
Standard	ARMv4	WIN CE 5.0	No	BrARMv4.exe
Intermec CV60	X86	WIN CE 4.2	No	BrX86.exe
Standard	X86	WIN CE 4.2	No	BrX86.exe
Standard	X86	WIN CE 5.0	No	BrX86.exe

Table 1: Device Setup Programs

Registration/Licensing

To run GSWBrowse on a Windows device you must register the software to have a valid license. Each copy of GSWBrowse is licensed for a single instance to run on a single device. The software will operate with full functionality without registration for evaluation purposes; however you will be reminded that the instance of GSWBrowse is not running a valid license when starting GSWBrowse and periodically thereafter.



Figure 1: Licensed and Evaluation Diagram

Registration entails just a few steps that involve GSWBrowse generating a unique Product ID (alphanumeric sequence of characters) for the device and providing this Identification to Georgia SoftWorks so a **Serial Number** (alphanumeric sequence of characters) can be generated. Georgia SoftWorks provides the Serial Number required to complete the registration.

GSWBrowse offers multiple registration techniques in order to provide flexibility and convenience for a variety of environments. Regardless of the technique, a Product ID is used to generate a Serial Number that is used for the registration.

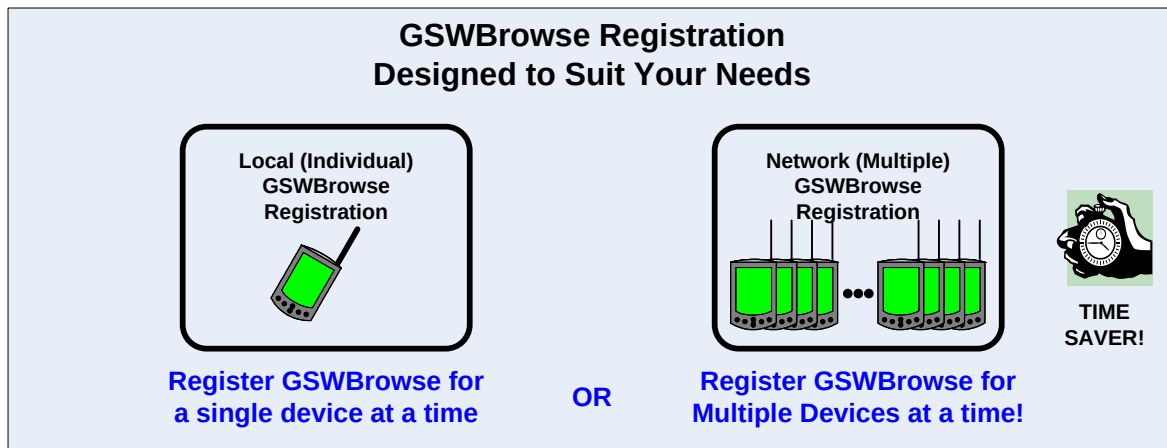


Figure 2: GSWBrowse Registration Techniques

Individual (Local License) GSWBrowse Registration

1. Run the Registration utility (RegUser) by selecting Licensing menu item as shown below by using the Menu Bar. **View -> Licensing**

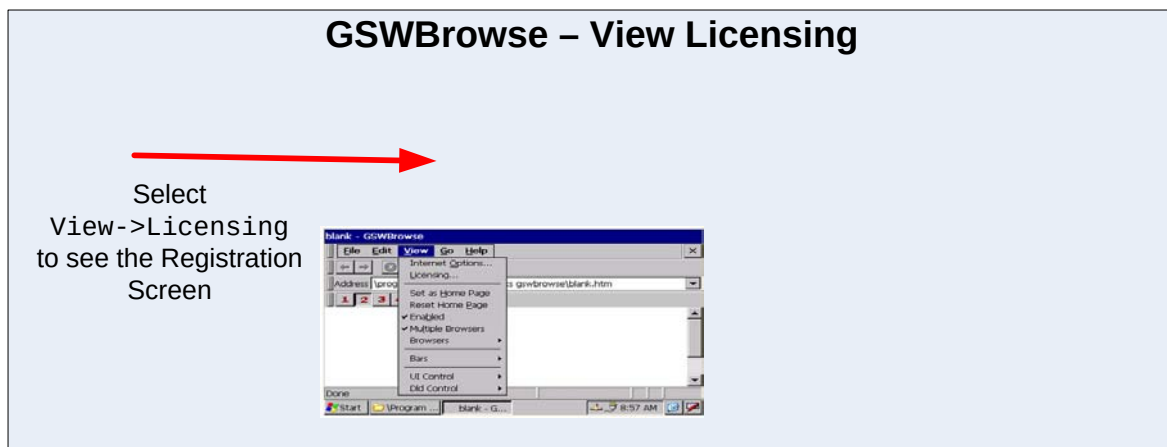


Figure 3: Menu Bar View -> Licensing

This opens the Registration form dialog as show in Figure 4 below.

GSWBrowse Registration

GSW Product Name for Registration

GSW Product Identification Number

GSW RegUser

GSW Product information

Name: Ver:

ID:

Registration information

Expiration date:

Free updates until:

Parameter:

Customer name

GSW Version Number

Figure 4: Registration Dialog

Certain GSW Product information is automatically filled by GSWBrowse. This includes the Name, version and Product Identification number.

Step 1: Complete the Customer Name Field. Simply enter the customer name in this field. This helps in verifying the registration request.

Registration – RegUser Dialog

STEP 1

Enter the name of your company to aid in verifying the registration request

GSW RegUser

GSW Product information

Name: Ver:

ID:

Registration information

Expiration date:

Free updates until:

Parameter:

Customer name

Click "Save..." to save it to a file that can be sent to GSW

STEP 2

Figure 5: Registration Dialog - Customer Name and Save

Step 2: Click Save and save the registration information to the text file named `register.txt`



The registration information must be provided to Georgia SoftWorks to obtain the Serial Number. Several methods are available for your convenience. The preferred method is via email; however fax or telephone can be also used.

EMAIL

Save the information to a file and email it to Georgia SoftWorks - *Preferred method.*

The registration information was saved when you clicked the **Save** button on the GSW RegUsr screen. Please email this file to Georgia SoftWorks registration@georgiasoftworks.com requesting registration.

FAX

Please print the register.txt file that was saved and fax the request to Georgia SoftWorks - 706.265.1020

TELEPHONE

Once Georgia SoftWorks receives the information, we can generate a Serial Number on demand. We will reply back via Fax or email. You may close the registration program at this time. Georgia SoftWorks - 706.265.1018

Apply the Serial Number

When the Serial Number is provided, it will need to be entered in the Serial Number field in the Registration Utility (RegUser) on the device. There are several options to accomplish the entry of the serial number.

- This can be done simply by entering the Serial Number from the keyboard of the device.

OR

- The **Register.txt** file received from GSW can be transferred to the device and the Serial Number copied/pasted from the file to the Serial Number field in the Registration utility.

OR

- If you are setup to use the Network (Multiple) Registration technique then take the LICENSE.XML file received from GSW and place it in the appropriate location on the license server. When GSWBrowse is launched it will automatically recognize the Serial Number.

Network (Multiple) Registration

In many environments, an administrator may be responsible for multiple devices ranging from a few to hundreds or even thousands. Recognizing the burden of having to register each device individually, Georgia SoftWorks has provided a network registration technique that substantially reduces the effort and time required. A high level overview describing the process follows.

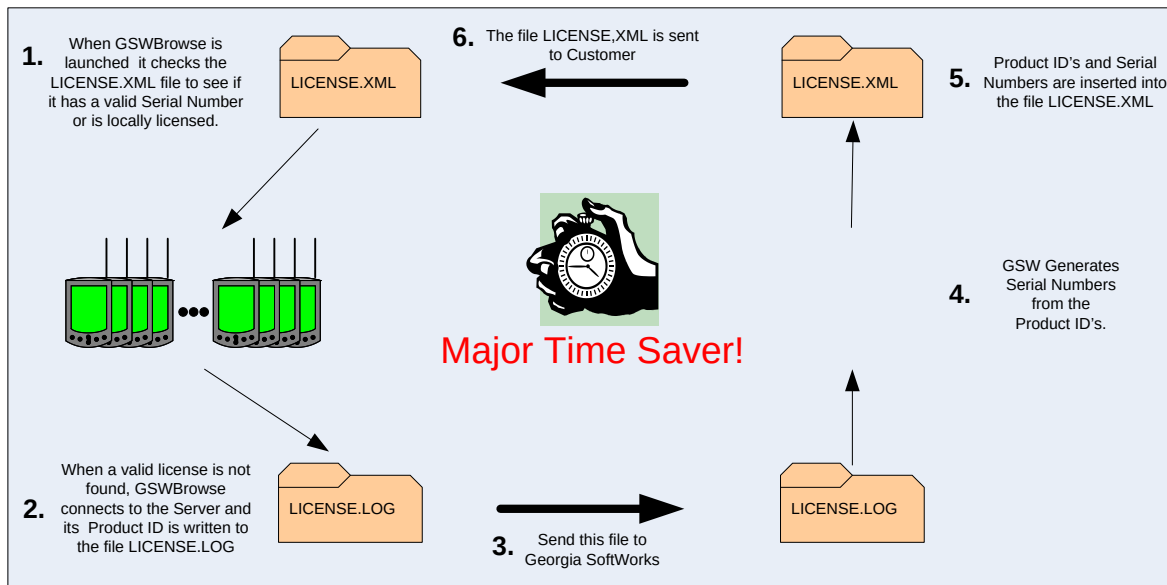


Figure 6: Network Registration – High Level Overview - Multiple GSWBrowse's

Each of the following items corresponds to a number in the above figure. **(1)** When a GSWBrowse is launched it checks the LICENSE.XML file to determine if a Serial Number is associated with its Product Identification. If there is a Serial Number associated with the Product ID, the license is validated. Otherwise GSWBrowse checks to see if it is locally licensed¹. **(2)** If there is not a Serial Number in the LICENSE.XML file and it is not locally licensed then the Product ID is sent to the server and written to the LICENSE.LOG file.

To register multiple browsers, simply launch GSWBrowse on each device and each Product Id will be written to the LICENSE.LOG file. **(3)** Next send the file to Georgia SoftWorks (email is preferred). **(4)** GSW will take each Product ID and generate an associated Serial Number. **(5)** The Product IDs and Serial Numbers will be placed in a file named LICENSE.XML and **(6)** sent back to the customer.

The customer takes the LICENSE.XML file and places it in the appropriate location on the server. The next time GSWBrowse is launched it will recognize the Serial Number associated with its Product ID and validate the license. Alternatively, the Serial Numbers can be manually entered in each Browser.

¹ Please see page 5 for details about Locally Licensed registration.



Figure 7 is a visual overview of the files needed for setup. *One* file is needed on the *client* and *three* files are needed on the *License Server*. Each of the files is identified in the yellow star in the figure below. Working examples of each file can be found in the IMPORTANT FILES section on [page 244](#).

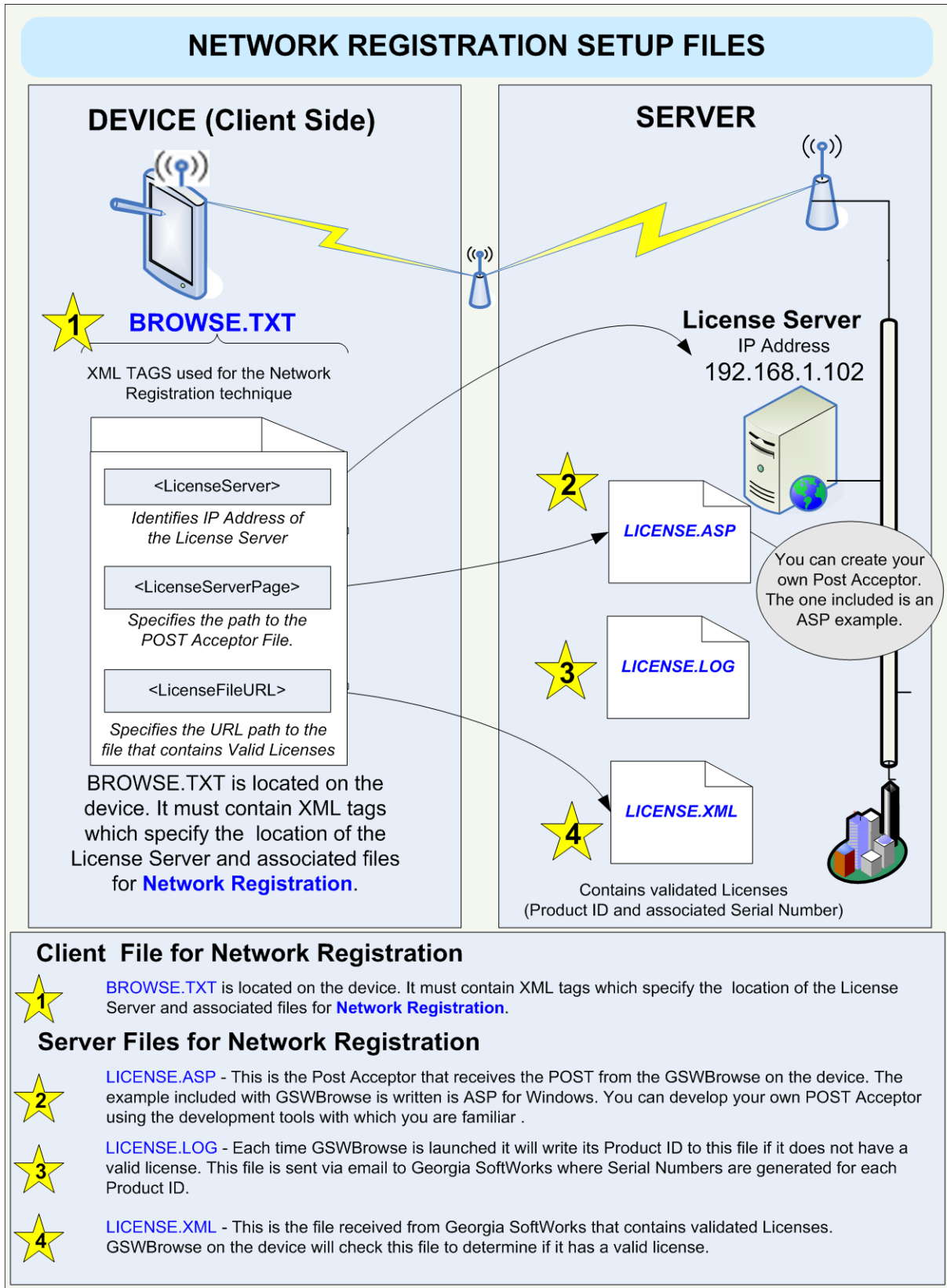


Figure 7: Network Registration Setup File Diagram



Network Registration Setup

Implementing the Network Registration entails a few configuration steps.

1. Modify the client side in the file GSWBROWSE.TXT so it has the correct paths to the Server with the Licensing information. GSWBROWSE.TXT is *installed on the Device* in its persistent folder. The persistent folder location is device specific. Please review the device documentation to determine the location of its persistent folder.

`<device persistent folder>/Georgia SoftWorks GSWBrowse/`

2. Launch GSWBrowse on each device. This causes each GSWBrowse to generate and send its Product ID to the LICENSE.LOG file on the server.
3. Email the LICENSE.LOG file to registration@georgiasoftworks.com
4. Georgia SoftWorks will validate and generate Serial Numbers for each Product ID and email a LICENSE.XML back.
5. Place the file LICENSE.XML in the appropriate folder (on the server) identified in the GSWBrowse.txt file (on the device). The next time GSWBrowse is launched it will identify that a Serial Number is associated with its Product ID. GSWBrowse will validate the Serial Number and upon success will be licensed.

GSWBROWSE.TXT

This is a client side file that provides GSWBrowse with configuration information. Some of the information contained in GSWBROWSE.TXT is used to identify the location of specific files on your web server. An example of GSWBROWSE.TXT can be viewed on [page 244](#).²

License Server

The location of the server where the License (Registration) files reside. The XML Tag `<LicenseServer>` is used to specify the IP address of the computer with the Licensing files. An entry such as the one below should be entered in the file.

Syntax

`<LicenseServer>IP Address of License Server</LicenseServer>`

Example

`<LicenseServer>192.168.1.102</LicenseServer>`

(Of course put your server's IP address between the License Server Tags, not the one in our example)

² Note: GSWBrowse can also contain XML tags which can specify other GSWBrowse configuration. See page 228 for details



License Server Page

The path to the *Post Acceptor* file. XML tag `<LicenseServerPage>` is used to specify the path to the Post Acceptor file. An entry such as the one below should be entered in the file.

Syntax

```
<LicenseServerPage>Path to POST Acceptor File</LicenseServerPage>
```

Example

```
<LicenseServerPage>/gswbrowse/licensing/license.asp</LicenseServerPage>
```

The included example ([page 245](#)) is a working example of a Post Acceptor ASP file. Those familiar with POST Acceptor files can write your own custom acceptor file using your favorite development tools.

License Server Port

Optional XML tag specifies the HTTP port where the data is posted.

Syntax

```
<LicenseServerPort>PORT NUMBER</LicenseServerPort>
```

where PORT NUMBER is a decimal value for the HTTP Port.

Default: 80, Port 80 is the default value. Unless a different port should be used, this XML tag can be omitted.

Example

```
<LicenseServerPort>81</LicenseServerPort>
```

License File URL

The XML tag that specifies the path to the XML file which contains the Product ID and Serial Number associations for registrations sent back by Georgia SoftWorks.

Syntax

```
<LicenseFileURL>PATH to License File</LicenseFileURL>
```

Example

```
<LicenseFileURL>  
http://192.168.1.102/gswbrowse/licensing/license.xml  
</LicenseFileURL>
```

Evaluation License

Prior to registration that obtains a license, GSWBrowse runs in an evaluation mode. This mode is fully functional and is available in order to provide you with a “Try before you Buy” opportunity. GSWBrowse performs a number of checks to determine if a license is present.

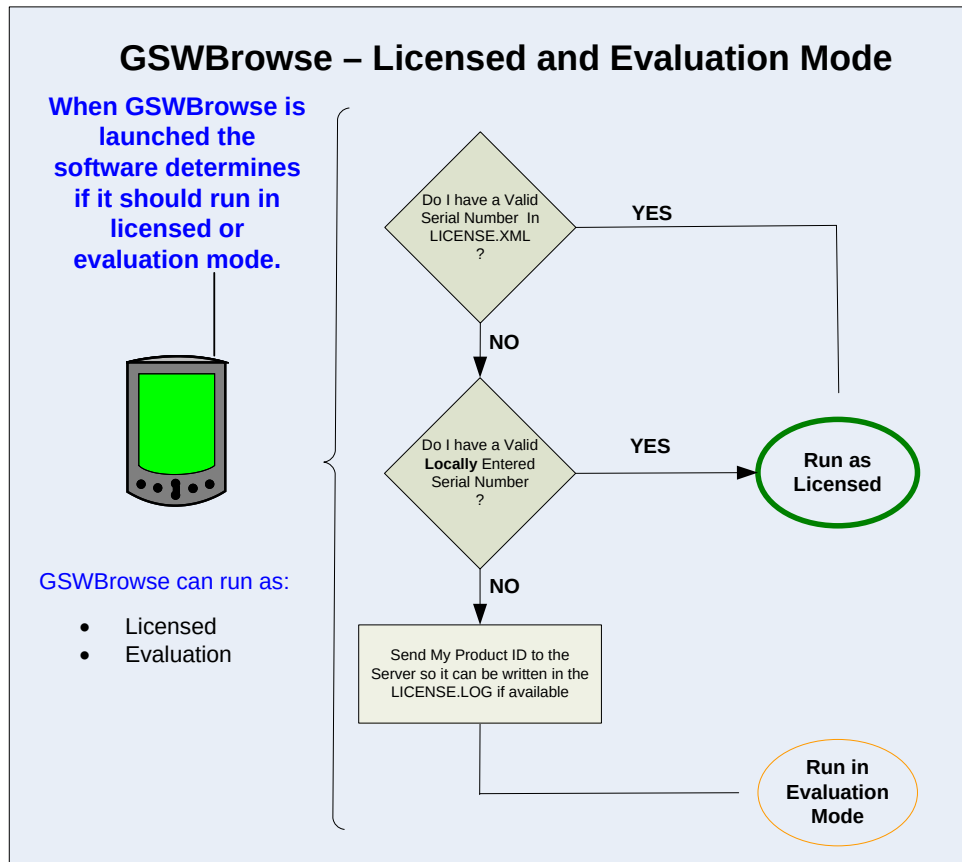


Figure 8: Licensed and Evaluation Flow Diagram

When GSWBrowse is launched, it examines LICENSE.XML to see if there is a Serial Number present. If so, then GSWBrowse runs as Licensed. Otherwise GSWBrowse checks to see if a valid Serial Number has been entered locally. If so, then GSWBrowse runs as Licensed. Otherwise GSWBrowse sends its Product ID to the server and it is inserted into the LICENSE.LOG file. This is done in preparation for the file being sent to Georgia SoftWorks where Serial Numbers can be generated.

The operational difference in evaluation mode is a periodical pop-up that serves as a reminder to obtain a license. Figure 9 is an example of the evaluation mode pop-up

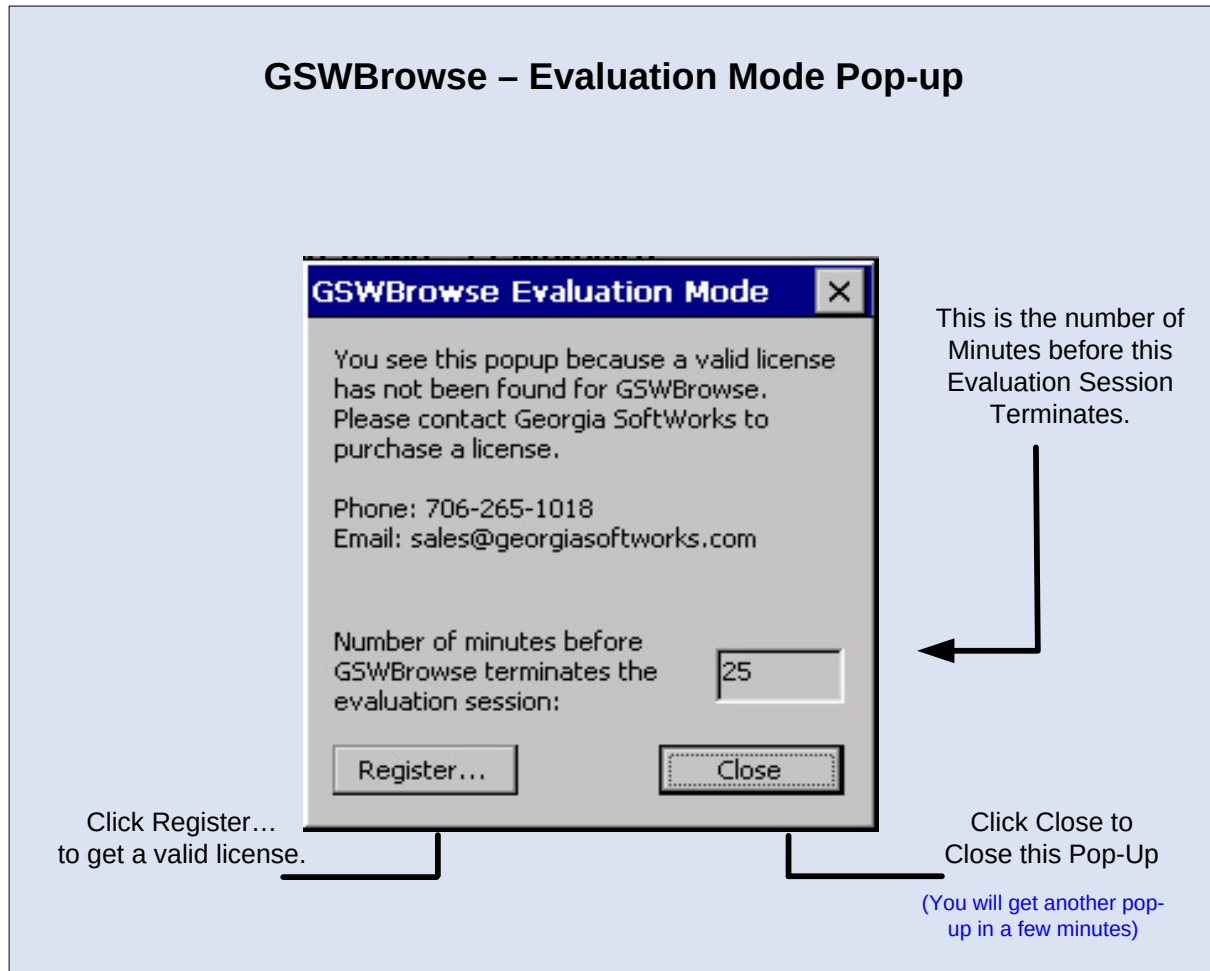


Figure 9: Evaluation Mode Pop-up

Thirty minutes is the amount of time allowed in evaluation mode before the session is terminated. After about five minutes a pop-up will occur with a reminder that this is an evaluation license along with the remaining time before this session is terminated. The frequency of reminders increase as the evaluation timer counts down. When zero, the session terminates and you must restart GSWBrowse.

GSWBrowse Layout

A typical GSWBrowse window is depicted below.



Figure 10: GSWBrowse Layout

The components that comprise the GSWBrowse browser are shown above.

The Caption, Tool, Address, Status and Task Bar operate as expected with a typical Windows Internet Explorer type browser.

The Menu Bar provides access to typical and extended commands. See [page 27](#) for details.

The Browser Tabs are displayed when more than one browser is enabled³. Five (5) Browsers are available. The Active Browser has the “pressed” appearance, browser one (1) in Figure 10.

The System Administrator can display or hide any or all “Bars”. This can help reduce unnecessary distractions while maximizing the Browser Content area.

Many common browser objects (Menu Bars, Tools) are available, but only the items providing GSW Browse enhanced RF Device and Data collection features are described in detail.

³ Please see page 19 for the Menu Item View commands to enable/disable browsers.

GSWBROWSE – Menu Bar – File - Security Level

Two security modes are available - Administrator and User mode. Administrator mode permits the authority to define the browser layout and items accessible in User mode. Control over the appearance of the browser as well as the scope of browser activities accessible to the user allows optimization of the display and operation for your specific environment.

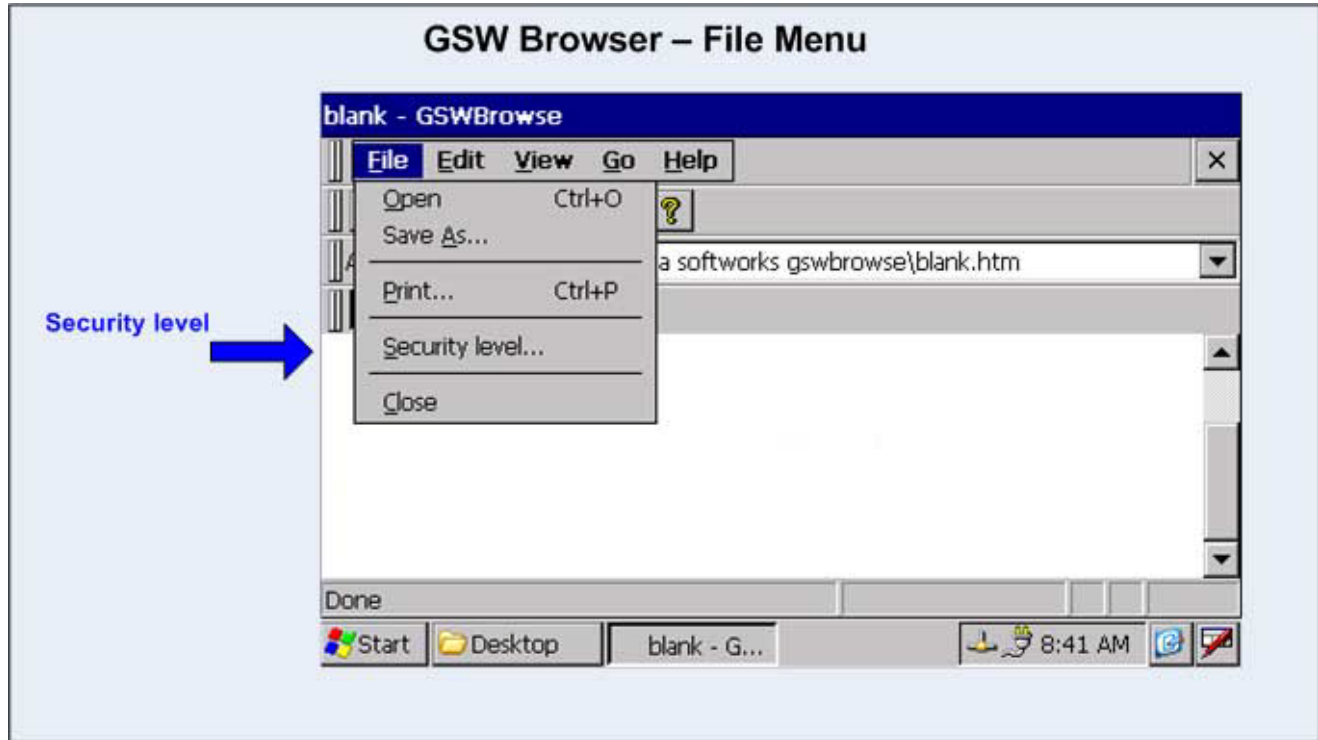


Figure 11: GSWBrowse - File->Menu->Security Mode

Typically, the administrator will enter the *Administrator Security mode*, configure the browser as it should appear to the user and then switch browser to *User Security mode* and exit. The next time the browser is started; it will be in User Security mode and will retain the configuration set by the Administrator. When the Security mode menu item is selected the following dialog is displayed.



Figure 12: Security Mode Dialog

Security Mode Dialog

Two methods are available to open the Security Mode Dialog. The first is to use the menu bar as shown in Figure 11

The second method to access the Security Mode Dialog is a keyboard sequence. Many times the menu bar may be hidden. This provides an alternate method of access. In User Mode this is the only method available to access the Security Mode Dialog. The key sequence is:

<CTRL> + <SHIFT> + S

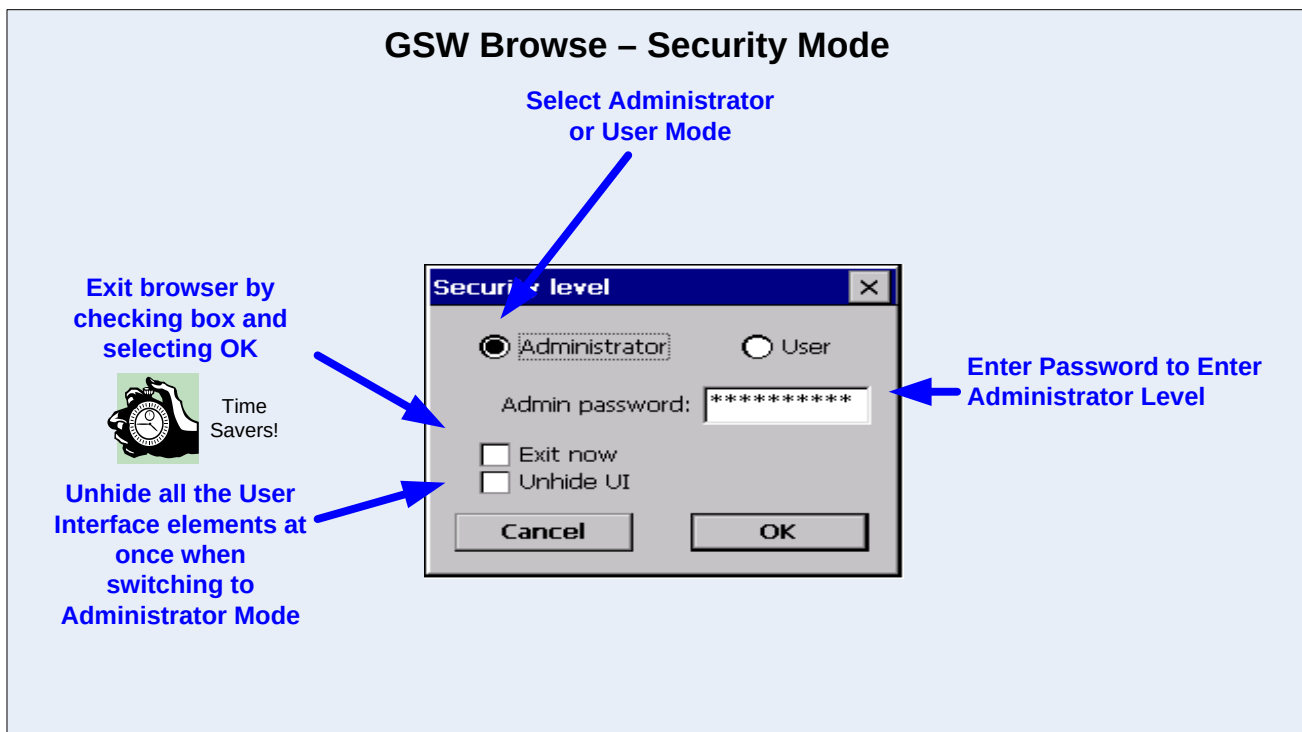


Figure 13: Administrator Security Mode

Administrator Security Mode

- Select Administrator
- Enter the password: This is GSW telephone number. 7062651018 Please note that the password only needs to be entered when switching from User mode to Administrator mode.
- To quickly "Unhide" and access all the User Interface elements, click Unhide UI. This is a time saver. Rather than enabling each User Interface element individually for each browser it unhides all UI elements for all browsers. If the UI elements are hidden, the administrator can still access menu items using ALT sequences. The ALT sequences to access hidden menus are only available in administrator mode.

- Make sure Exit Now is unchecked when entering Administrator mode.

In Administrator Mode, each browser can be configured to the desired layout for the user. This includes all the Bars (Status, Caption, Toolbar, Address, Menu and Taskbar), multiple or single browsers, and the Home page. When the Security Mode is switched back to User the layout will be exactly as configured.

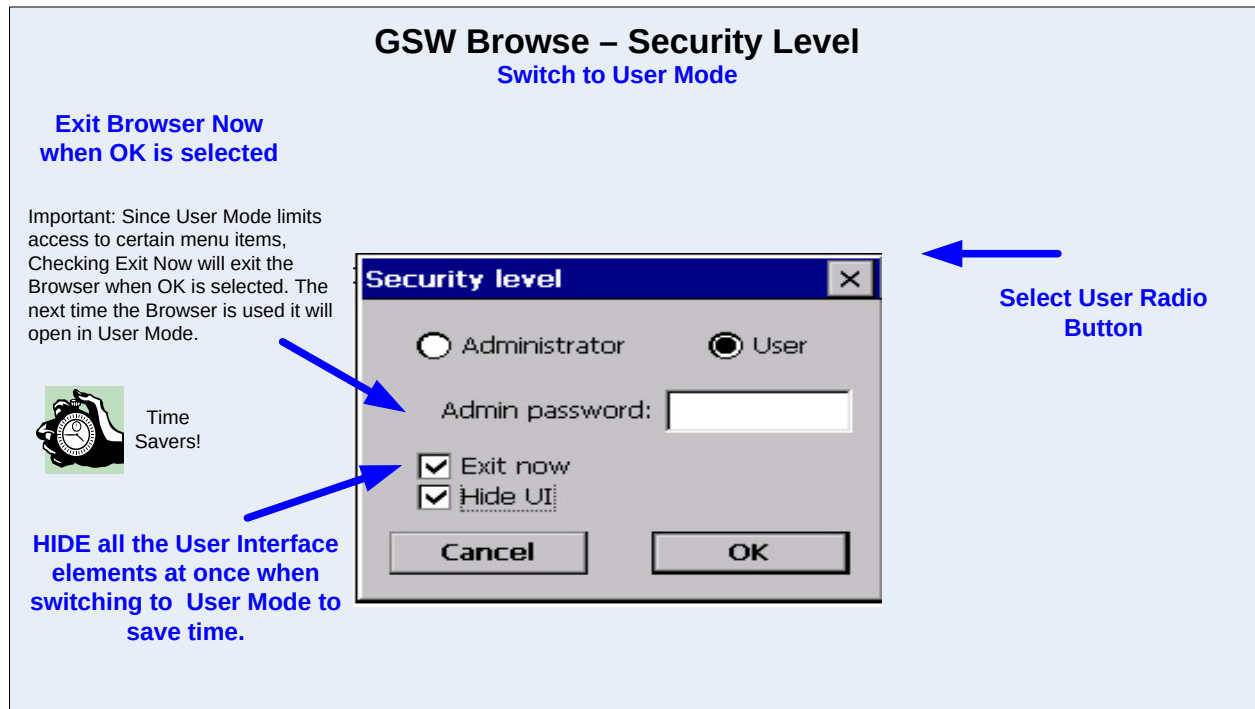


Figure 14: Switch to User Mode

User Security Mode

- Select User
- To quickly “hide” access to all User Interface elements at once, click Hide UI. This is a time saver, rather than having to hide each one individually. To display User Interface elements specifically defined in Administrator mode, leave Hide UI unchecked.
- When “Exit now” is checked, GSWBrowse will exit upon selecting of the OK button. This is important because it is the only way to exit the browser when the UI is hidden. If you forget to click the Exit Now button then re-enter the Security Mode using the key sequence technique to correct and exit.

Lock Down

The administrator can “Lock” the user from leaving the browser by disabling the Address Bar, the Menu Bar Taskbar and the Toolbar and then switching to User Mode. This typical and maximal lock down is accomplished by checking hide UI in the Security Dialog.



The administrator can determine the layout of the items available such as home page, menus, tool bars; status bars; caption bars; etc. removing distractions and increasing the display area. Once the layout is defined then the Administrator can “lock it” for User Mode.

The ability to perform the “Lock Down” is a major design goal of GSWBrowse.

GSWBROWSE – Menu Bar – Edit

The Edit | Printer Config.. item in the Menu bar is where the printers are configured. This includes local printers such as serial, Bluetooth and IrDA as well as network printers accessible to the client device. GSWBrowse allows each browser to have a separate printer configuration. This offers the flexibility to configure a different printer for each browser if desired.

Printer Configuration Overview

Select the browser that you want to specify a printer configuration. The left side of the figure below shows browser 1 as selected. Next select Edit | Printer Config... menu item and the dialog in Figure 16 is displayed.

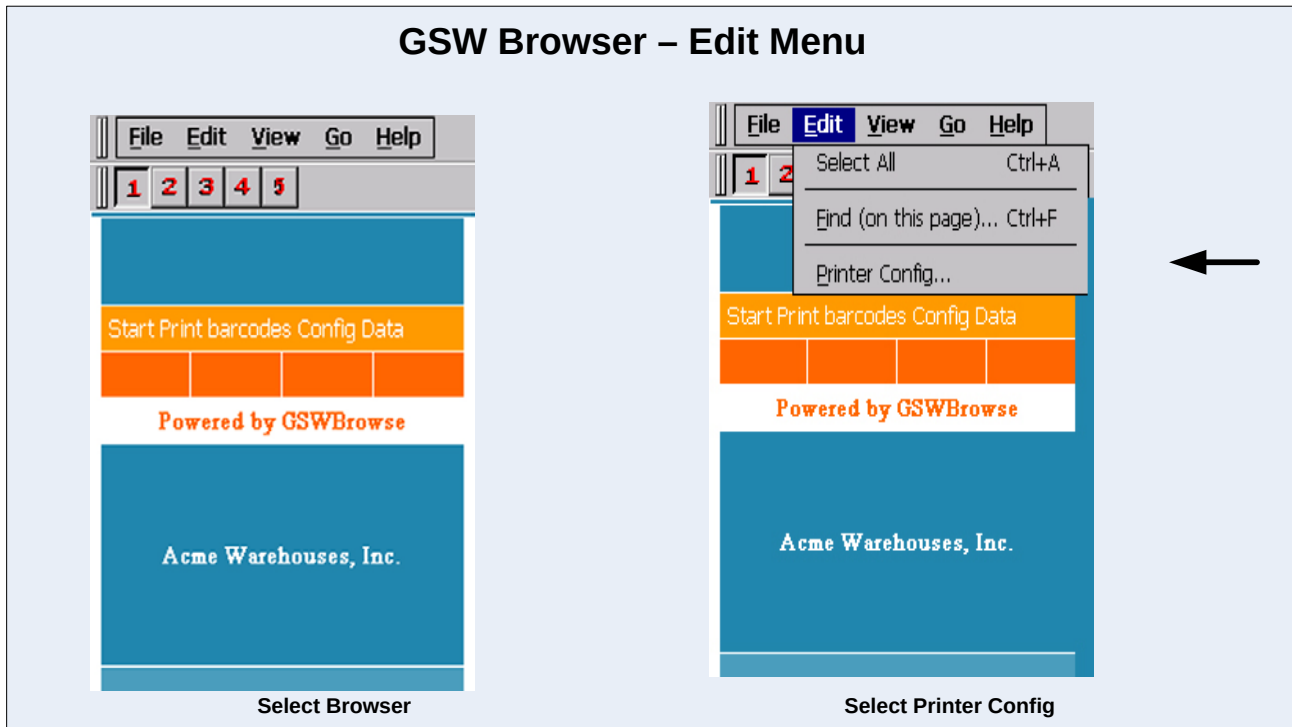


Figure 15: Menu bar - Edit (Printer Configuration)

The “Enable client side printing” checkbox specifies if this browser has printing enabled.

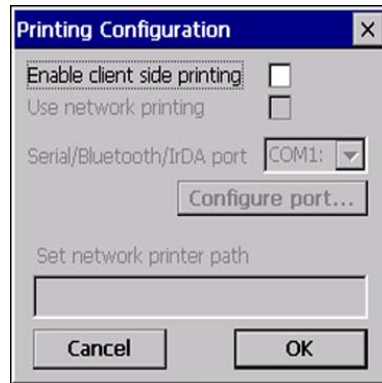


Figure 16: Printer Configuration - Enable Client Side Printing

A browser may have either a network printer enabled or a local (Serial/Bluetooth/IrDA) printer port enabled but not both. When the Printing Configuration is initially opened, the configuration options for Local (Serial/Bluetooth/IrDA) printer ports are available. To switch to network printing simply check the **Use network printing** checkbox and the network path entry box is enabled and while the Serial, Bluetooth, IrDA options are disabled.

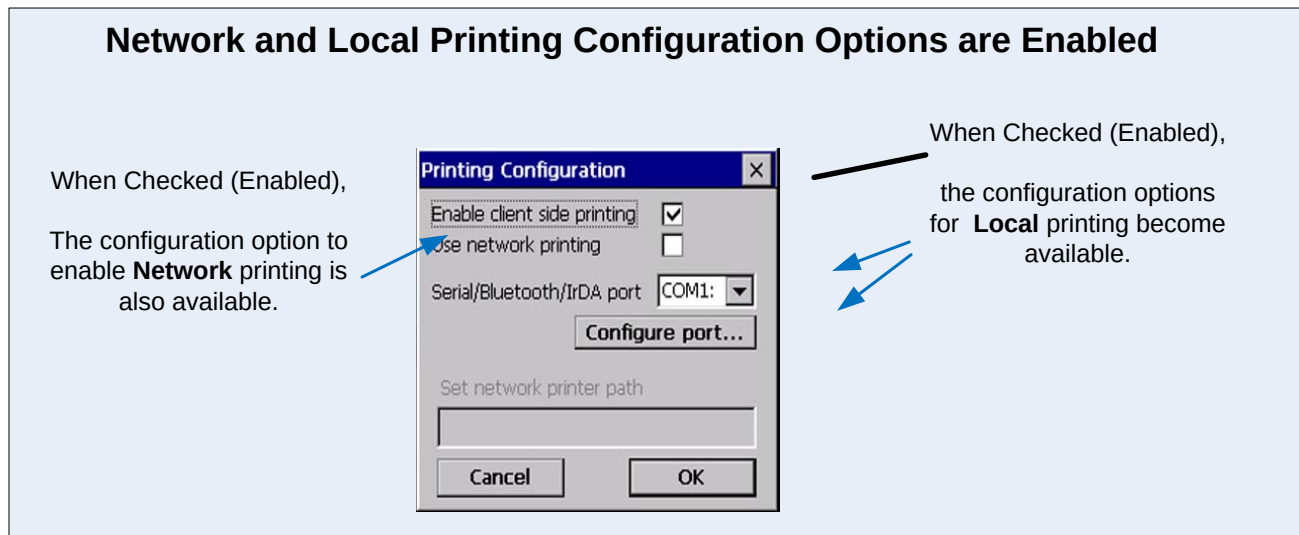


Figure 17: Network and Serial/Bluetooth/IrDA options enabled

Local (Serial/Bluetooth/IrDA)

Local printer configuration steps.

1. Select “Enable client side printing”
2. Make sure that “Use network printing” checkbox is NOT selected.
3. Choose the Communications Port
4. Configure the port to make sure the device and printer match.

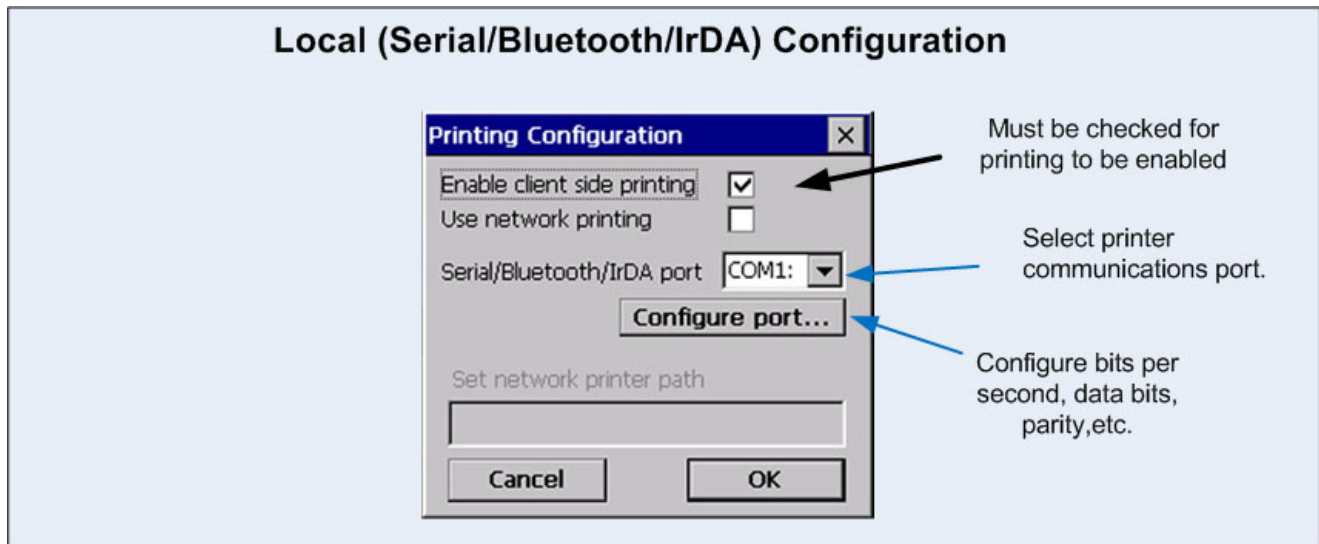


Figure 18: Enable Client Side Printing

Figure 19 shows details about the communications port and port properties options.

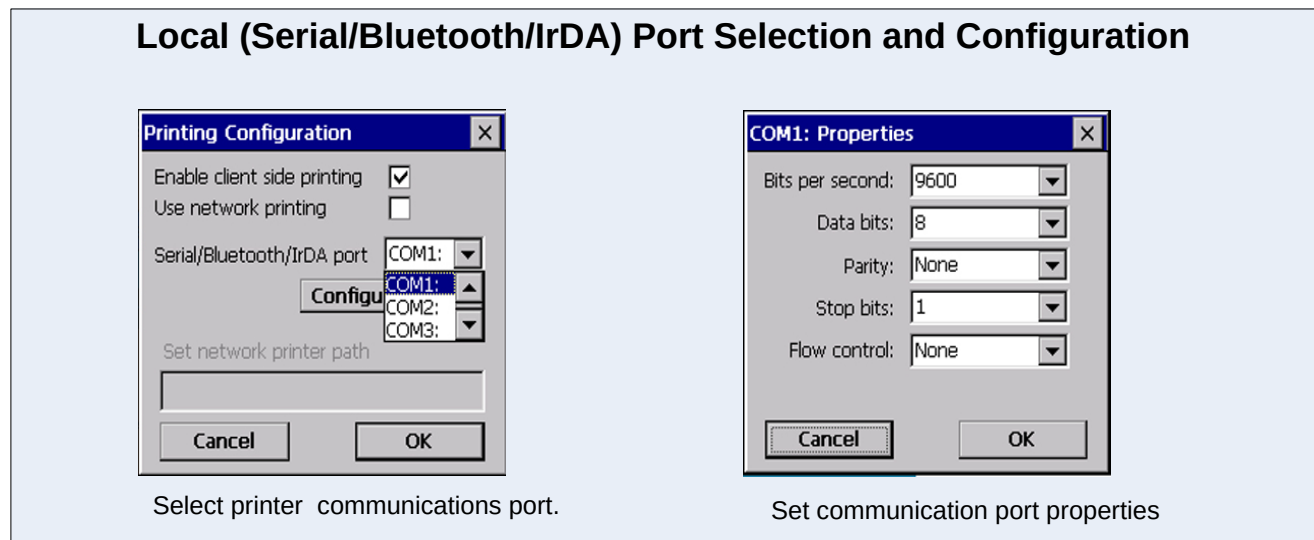


Figure 19: Local Printer port selection and configuration properties

Network Printer

Network printer configuration steps.

1. Select “Enable client side printing”
2. Select the “Use network printing” checkbox.
3. Enter the network printer path.
4. Click OK.

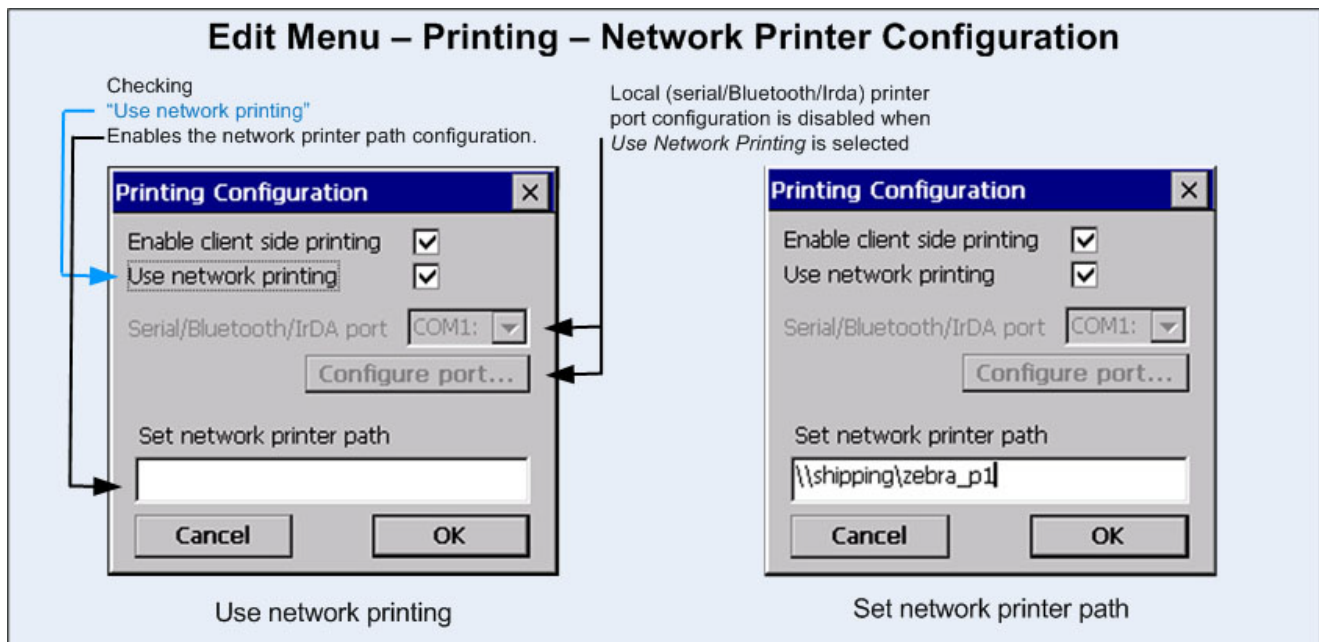


Figure 20: Printing - Network Printer Configuration

GSWBROWSE – Menu Bar – View

The View Menu bar is where the bulk of the GSW Browse layout is configured. Additionally, the GSWBrowse Licensing /Registration information is available through the View Menu item.

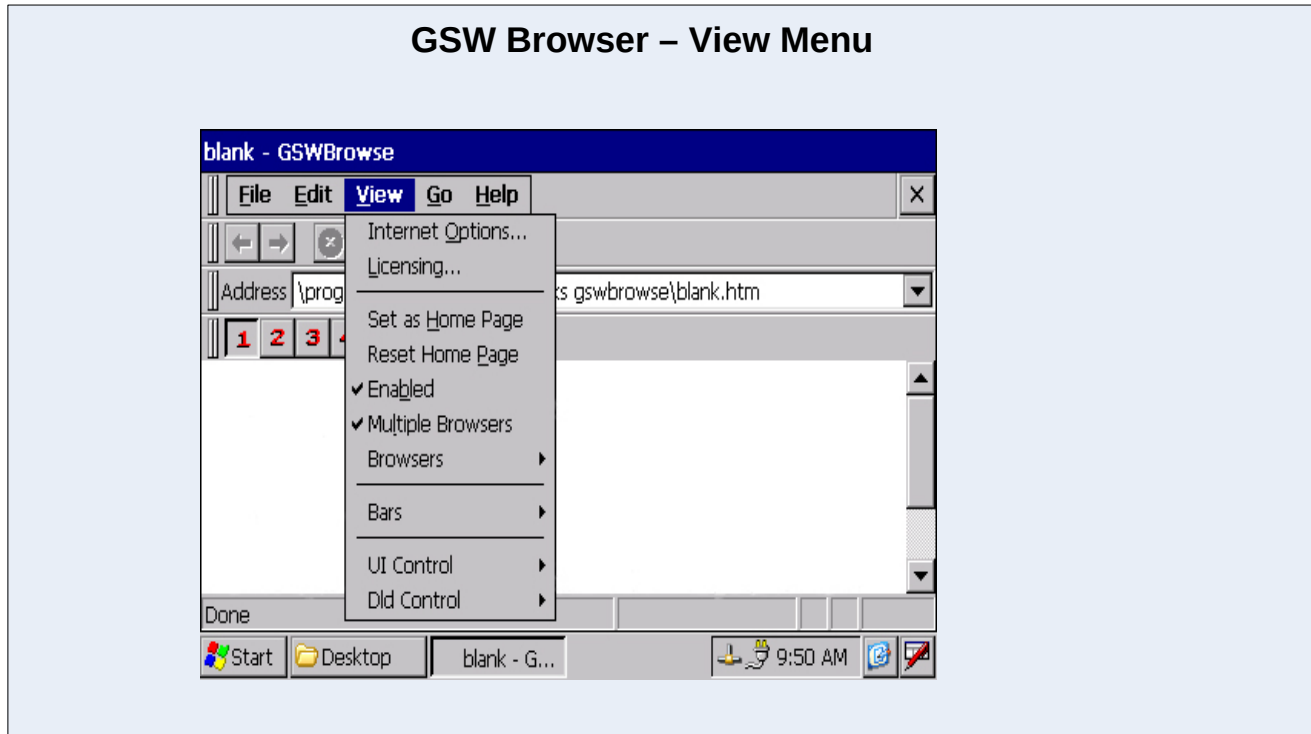


Figure 21: Menu Bar - View

The GSWBrowse enhanced RF device and data collection features in the View menu are:

Licensing

View/Modify Registration information. GSWBrowse must be registered to operate without the frequent reminder pop-up. Please see GSWBrowse Registration on [page 4](#). Default: **Admin Mode** only.

Set As Home Page

Navigate to the desired page and select “Set as Home Page” to make this the home page for the browser. Default: **Admin Mode** only

Reset Home Page

Removes (resets) link to the Home Page. This returns the Browser to the initial page as when initially installed. This can be used as a memory saving feature for memory critical devices. Default: **Admin Mode** only

Enabled

Enables or Disables the currently active (visible) browser. The User will not be able to access this browser when disabled. Default: **Admin Mode** only.

Multiple Browsers – Enabled (Checked) allows access to any of the five browsers that are enabled. Disabled (Unchecked) only allows access to a single browser. Default: **Admin Mode** only.

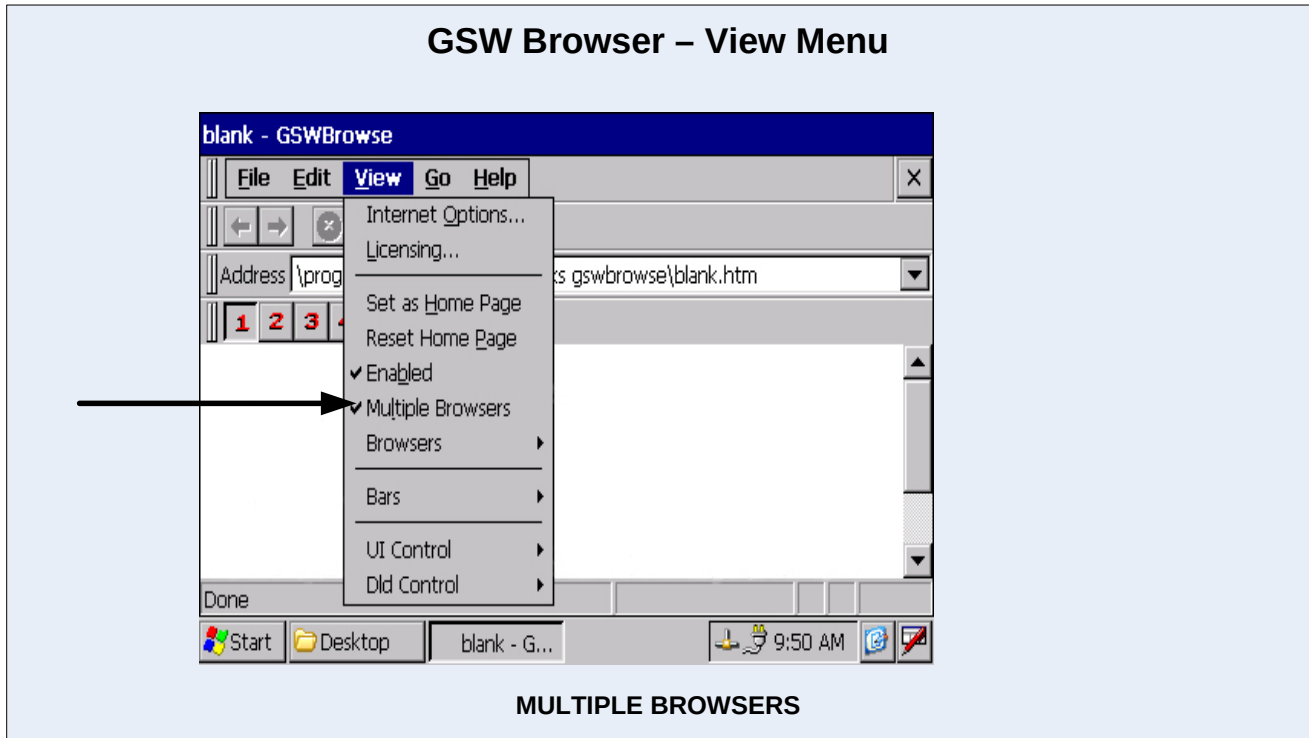


Figure 22: Multiple Browsers Configuration

When only a single browser is needed then disable (uncheck) Multiple Browsers. The Browser Tabs will no longer be displayed saving display area and unnecessary distraction.

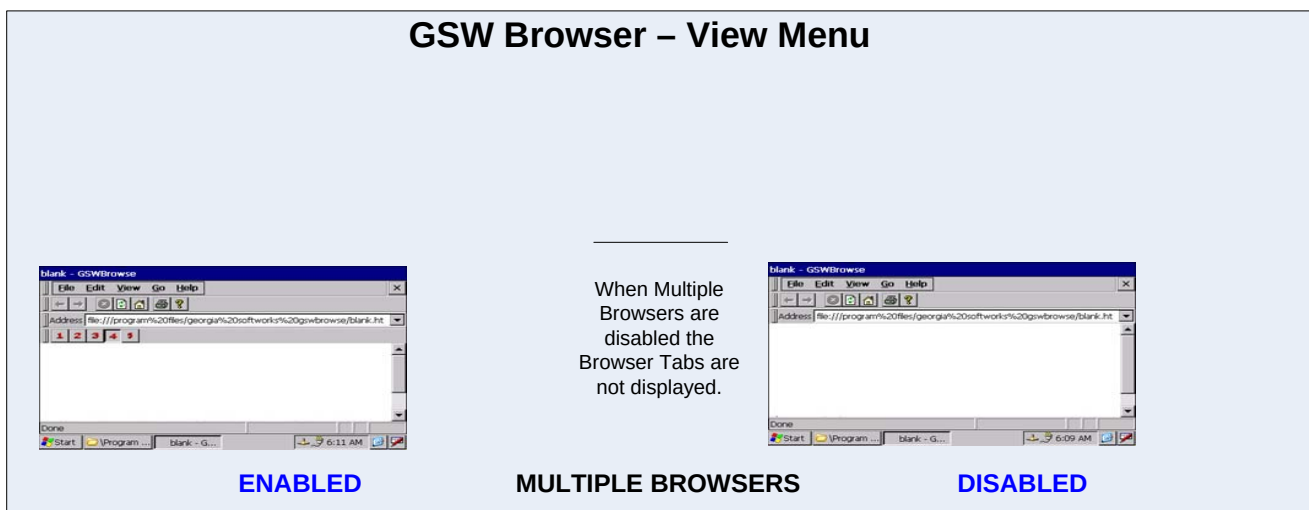


Figure 23: Multiple Web Browsers Enabled/Disabled

Browsers

Displays the current browser instance (noted by the check) and allows switching to another browser. Only enabled browsers can be selected in User Mode. Default: **Admin and User Mode**.

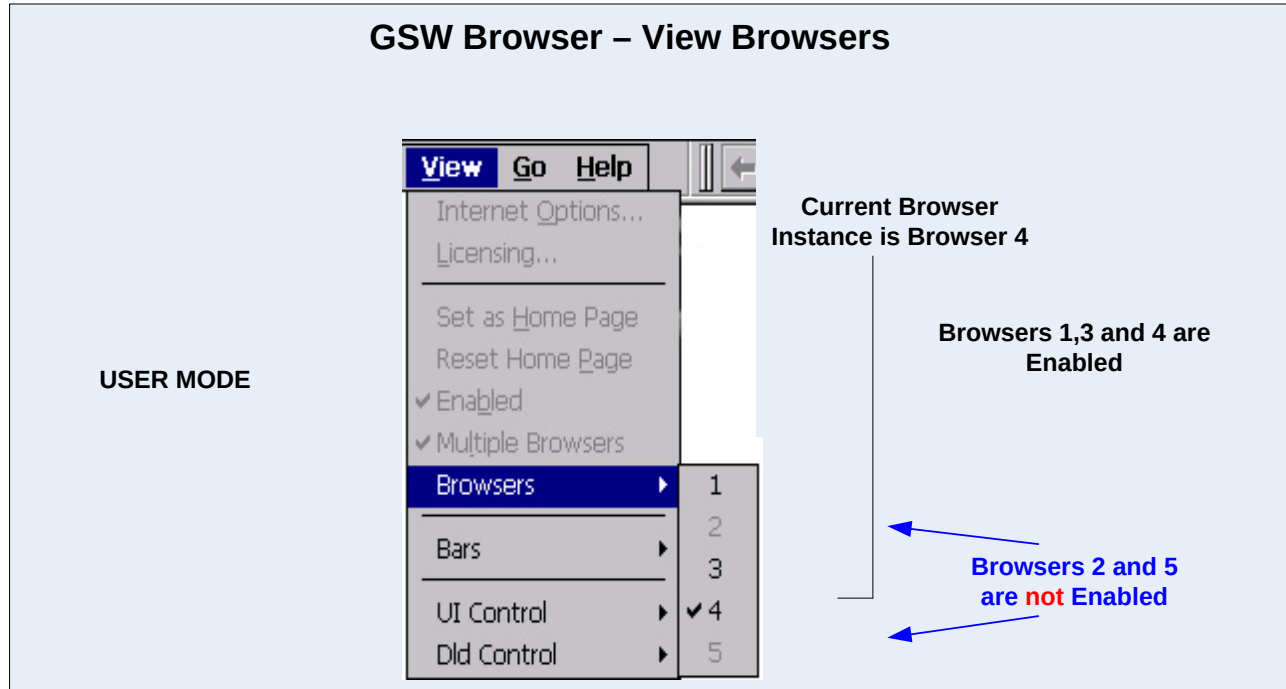


Figure 24: Select Browser Menu Item

The Browser Tabs offer the same information in a less intrusive format while offering a quick way to switch between Browser Instances.

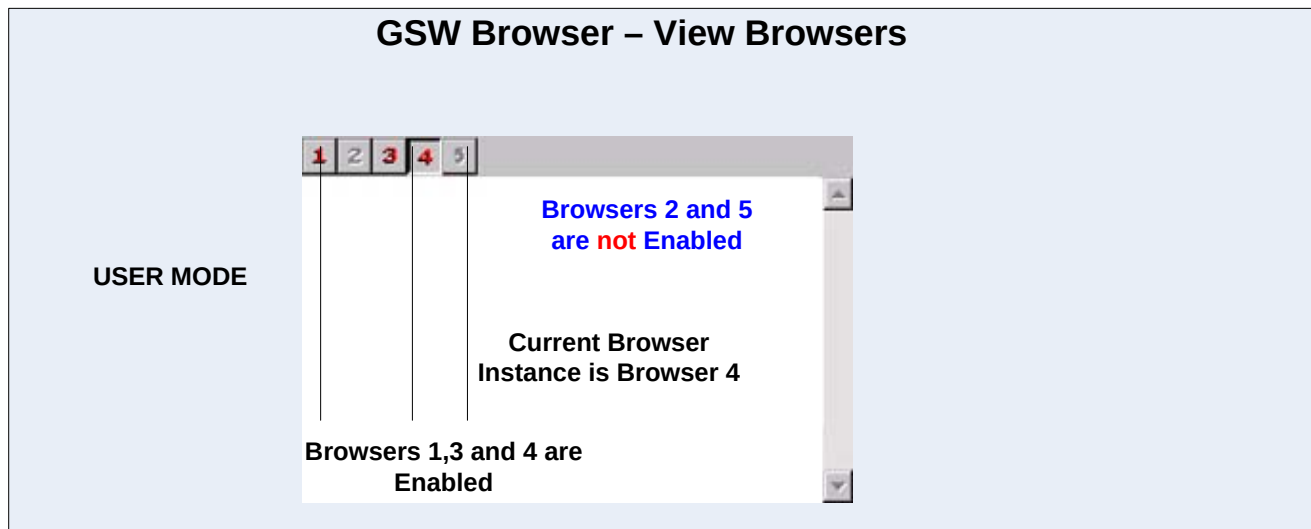


Figure 25: Browsers Tab

Bars

Enable/Disable and display the Bars available. Each browser's Bars can be individually configured. The menu selections in this section refer to the currently selected browser instance.

- Shows what Bars are enabled - **User Mode**.
- Enables/Disables the various Bars – **Admin Mode**

If specific bars are not required for your application, the administrator may opt to disable the bars that are not needed. Often the display on barcode scanners and data collection devices is limited and it is important to maximize the content display area. Browser bars and browser tabs may be moved in the common manner.

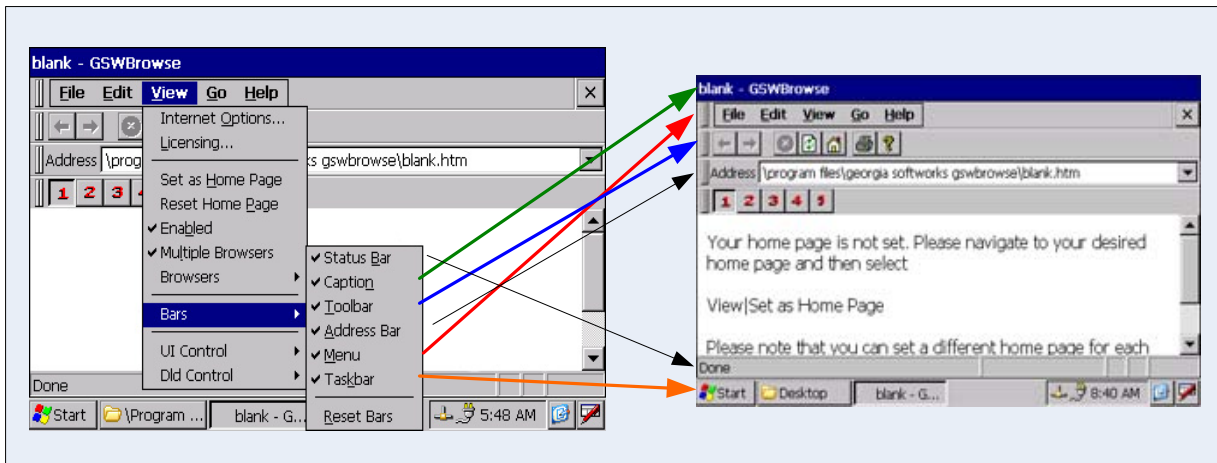


Figure 26: Diagram of Browser Bars

Refer back to Figure 10 to view another diagram which shows the Bars. Please note that the Scroll Bars and the Browser tabs are not configured here. Please see appropriate section for configuration of **Scroll Bars** and **Browser tabs**.

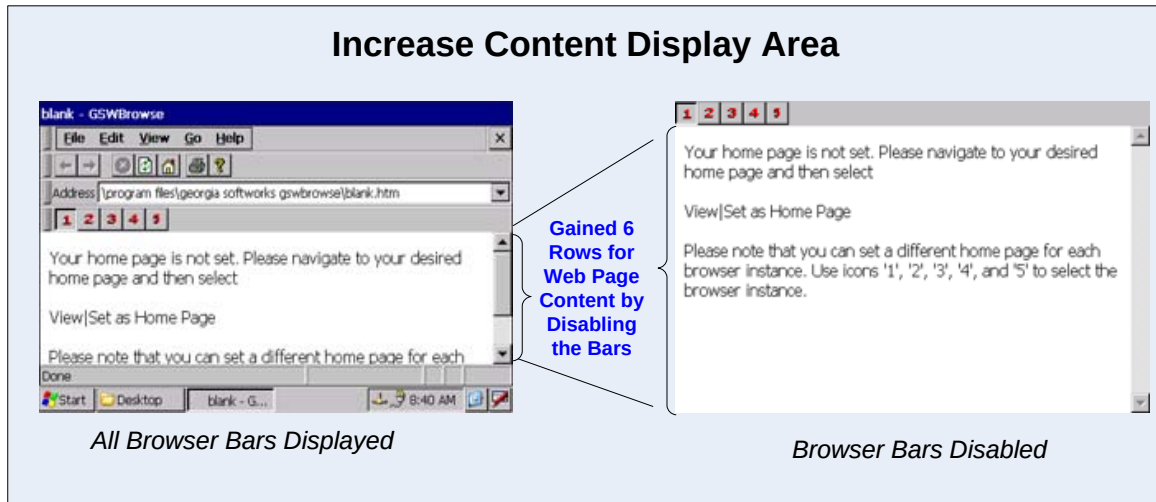


Figure 27: Maximize Content Area - Hide Bars

Still need more content area? Eliminate the Browser Tabs if they are not needed; leave just the content area and scroll bars. Still need more room, remove the scroll bars as shown in the figure below. GSW Browse is capable of being configured in order to meet a wide range of specific display requirements.

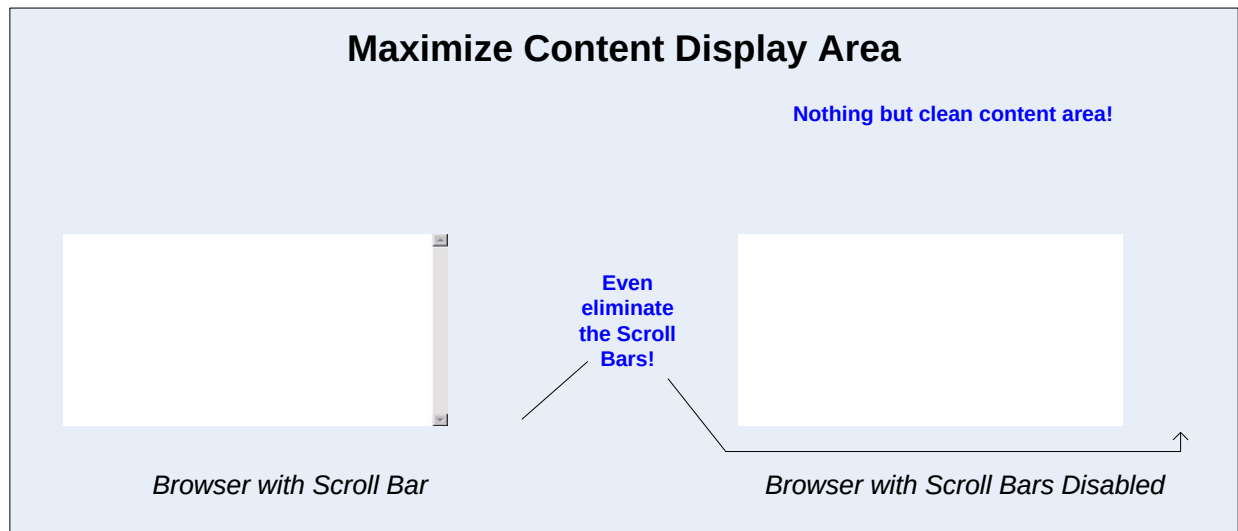


Figure 28: Maximize Content Area

See [page 32](#) for configuration to remove the scroll bars.

UI Control – User Interface Control

GSWBrowse supports configuration of the enabling and disabling of various UI behaviors.

Admin Mode.

The Administrator may determine that Users should have certain browser user interface capabilities while restricting other ones. The UI Control allows the Administrator to enable/disable a wide range of interface operations on a per browser instance⁴ basis. This includes the selection of text, context menu control as well as control over browser display properties such as scroll bars and 3D borders.

The Administrator specifies what UI Control features are available to the user by configuring the UI Control settings.

The first group of UI Control settings provides the administrator ability to select the context menus that are displayed in the browser as a result of a right-click. The second group of UI Control settings provides text selection, 3D Border and Scrollbar control.

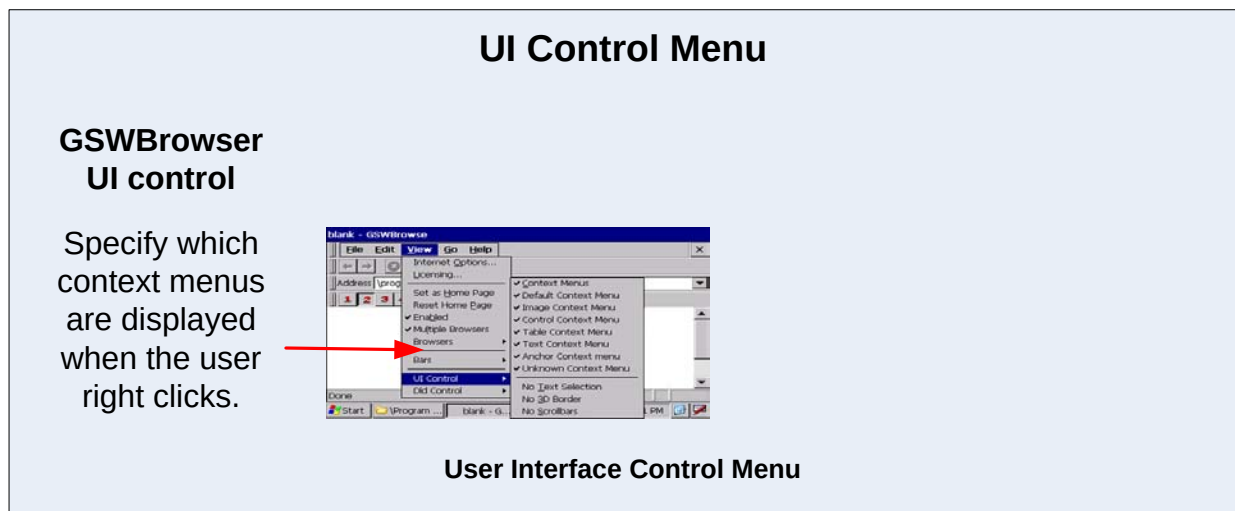


Figure 29: UI Control Menu

⁴ Each of the 5 browser instances can have different settings.

CONTEXT MENUS OVERVIEW

Typically when a user right clicks (or taps and holds when using a stylus) in a browser a context menu is displayed providing actions commonly associated with the “context” of the object right-clicked. This may not be appropriate or useful on RF Device or Data Collection terminals.

The left side of Figure 30 shows an example of a *text context menu* that is displayed when the user selects the word “*different*” in the text and right-clicks. The menu with Copy, Select All, and Print appears.

When the “text context menu” is disabled, the right side of the Figure 30 shows the new behavior. Even if the user selects text, they will not get a context menu when they right-click.

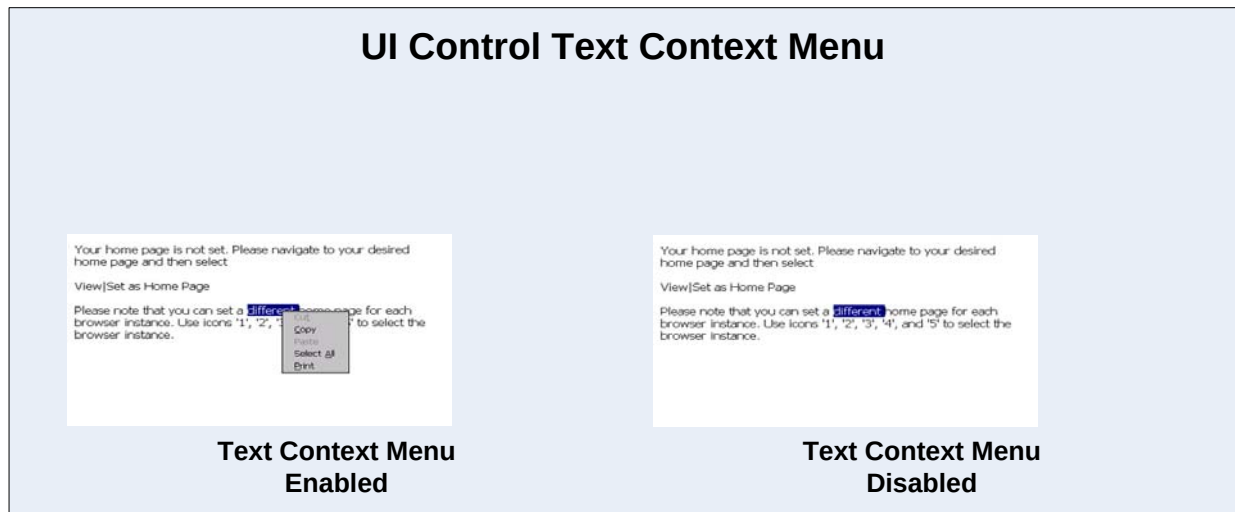


Figure 30: UI Control - Text Context Menu

CONTEXT MENUS



Time
Saver!

Enable/Disable **ALL** context menus. A quick way to *disable* the display of all context menus for GSWBrowse.

Default: **Enabled** - Allow context menu operation.

DEFAULT CONTEXT MENU

Enable/Disable the default context menu.

The Default context menu is displayed when a right mouse click or equivalent is performed on an empty area of the browser content.

Default: **Enabled** - Allow context menu operation.

IMAGE CONTEXT MENU

Enable/Disable the context menu for Images.



The image context menu is displayed when a right mouse click or equivalent is performed on an image area of the browser content.

Default: **Enabled** - Allow Image context menu operation.

CONTROL CONTEXT MENU

Enable/Disable the context menu for custom controls.

The control context menu is displayed when a right mouse click or equivalent is performed on a control area of the browser content.

Default: **Enabled** - Allow Control context menu operation.

TABLE CONTEXT MENU

Enable/Disable the context menu for Tables.

The table context menu is displayed when a right mouse click or equivalent is performed on the table area of the browser content.

Default: **Enabled** - Allow Table context menu operation.

TEXT CONTEXT MENU

Enable/Disable the context menu for Text.

The text context menu is displayed when text is selected followed by a right mouse click or equivalent.

Default: **Enabled** - Allow text context menu operation.

ANCHOR CONTEXT MENU

Enable/Disable the context menu for Anchors (links only). This does not include images or image maps.

Default: **Enabled** - Allow anchor context menu operation.

UNKNOWN CONTEXT MENU

Enable/Disable the context menu when a right click is performed on something other than any of the above.

Default: **Enabled** - Allow Unknown context menu operation.

NO TEXT SELECTION

Enable/Disable the ability for the user to select text.



Default: Enabled - Allow text selection.

When checked: Do not allow text selection in the browser content window.

NO 3D BORDER

Enable/Disable the display of 3D borders for the browser (when supported).

Default: Enabled - Allow 3D borders.

When checked: Do not allow the display of 3D borders for the browser.

NO SCROLLBARS

GSWBrowse displays vertical and horizontal scroll bars to access content outside of viewing area of display. Scroll bars may be disabled to obtain additional content area. If scroll bars are disabled then the application developer should be careful to ensure that all content will fit on the available area. In most cases using scroll bars is not a good design practice for data collection devices.

Default: Enabled – Vertical and Horizontal Scroll Bars.

When checked: Do not display scroll bars.

Dld Control – Download Control

The Administrator may determine certain browser downloads from the server should be allowed while restricting others. The **Dld** Control allows the Administrator to enable/disable different download capabilities ranging from sound, video to Java Applets and ActiveX controls. – **Admin Mode**



Figure 31: Dld (Download Control)

Each of the GSWBrowse Dld Control options is listed below. Please be careful to note whether a checked item specifies enabled or disabled. GSWBrowse patterned the terminology after Microsoft common programming terminology to maintain a familiar feel for administrators acquainted with its programming.

IMAGES

Default: Enabled - Allow the download and display of images.

When not checked: Do not allow images to be downloaded from the server.

VIDEOS

Default: Enabled - Allow the download and playing of videos.

When not checked: Do not allow videos to be downloaded and played from the server.

SOUNDS

Default: Enabled - Allow the download and playing of sound.

When not checked: Do not allow sounds to be downloaded from the server.

NO SCRIPT EXECUTION

Default: Disabled – Allow execution of client side scripts.



When checked: Do not allow execution of client side scripts.

NO JAVA APPLET

Default: Disabled – Allow execution of Java Applets.

When checked: Do not allow execution of Java Applets.

NO EXECUTION OF ACTIVEX CONTROLS

Default: Disabled – Allow the download and execution of ActiveX controls from the server.

When checked: Do not allow ActiveX controls to be downloaded and executed from the server.

NO CLIENT PULL

Default: Disabled – Client Pull Operations will be performed.

When checked: Do not allow Client Pull Operations to be performed.

NO FRAMES DOWNLOAD

Default: Disabled – Allow the download of frames.

When checked: Do not allow download of frames but download and parse the frameset page. GSWBrowse will ignore the frameset and render a no frames tag.

FORCE OFFLINE

Default: Disabled – GSWBrowse operates in On-Line mode.

When checked: Force GSWBrowse to operate in Off-Line mode.



Developers Guide

Overview

GSWBrowse offers the web page developer⁵ a variety of options to accomplish development tasks on the device. GSWBrowse provides straightforward and easy to understand techniques and syntax to interface with the device and browser. When supported by the device the GSWBrowse development interface can perform numerous client side actions/functions. This includes but is not limited to obtaining the device battery status, controlling leds, beeps, hooking the keyboard, performing scanner operations, determining RF signal strength, invoking GSWBrowse menu items and much more.

GSWBrowse developer features are available from JavaScript and HTML and seamlessly integrate with your web page development. GSW provides two ways to access these features. One is through GSW Action Commands and the other is through the window.external programming extensions.

GSW Action Codes

GSW Action codes are composed of the actual command name followed by optional parameters. Frequently throughout this document the parameters are referred to as “*content*”. The format and value of the content (parameters) are dependent on the command.

The content can range from simple numbers, time and dates to JavaScript functions and URLs. Additionally content can be used to pass values embedded in a URL as if a form was SUBMITTED, to be handled by server side code such as CGI, ASP, etc.

For example, if you wanted to set the time on the device to 19:20 (7:20pm) you would use the command – **gswbrowse_SetTime** and set the content to **19:20**.

GSWBrowse terms the combination of the command and parameter (if any) as a Georgia SoftWorks Action Code (gswac). Multiple programming techniques can be used to invoke commands and pass the parameters.

⁵ A certain level of programming skill is presumed for the developer that includes basic knowledge of HTML and JavaScript.

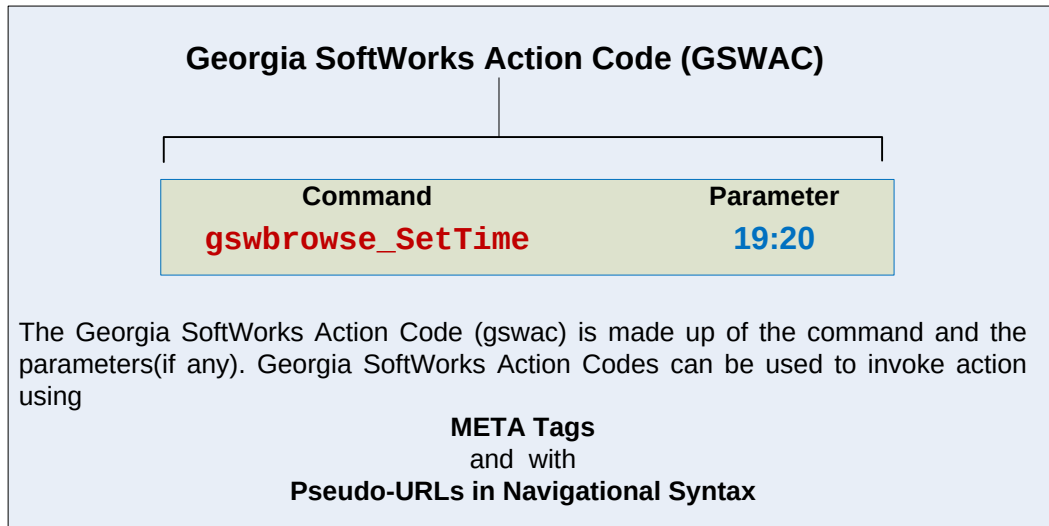


Figure 32: GSW Commands Action Code Definition Diagram

GSW Action Codes can be invoked when the page loading is complete, as well as based on other common events (E.g. form, button, mouse, etc.) or when a programming logic condition is satisfied.

window.external extensions

GSW uses a standard JavaScript mechanism for providing vendor extensions and provides a number of objects with their own properties and methods.

In summary

HTML Meta Tags, HTML Anchor tags, specialized functions, object properties and methods through proprietary GSW extensions to the “window.external” model are some of the techniques available to the developer to take advantage of extensions provided by GSW in GSWBrowse. Each technique requires a specific syntax to specify the command and parameters or objects and its properties or methods.

We will review these in the following sections.

GSW HTML Meta Tags Overview

GSWBrowse offers programmatic access to its own functionality in addition to the device's via META tags.

Device Specific Meta Tags – Certain manufacturers of devices along with different operating system versions have specific Meta tags to accomplish tasks. GSWBrowse supports as many as possible.

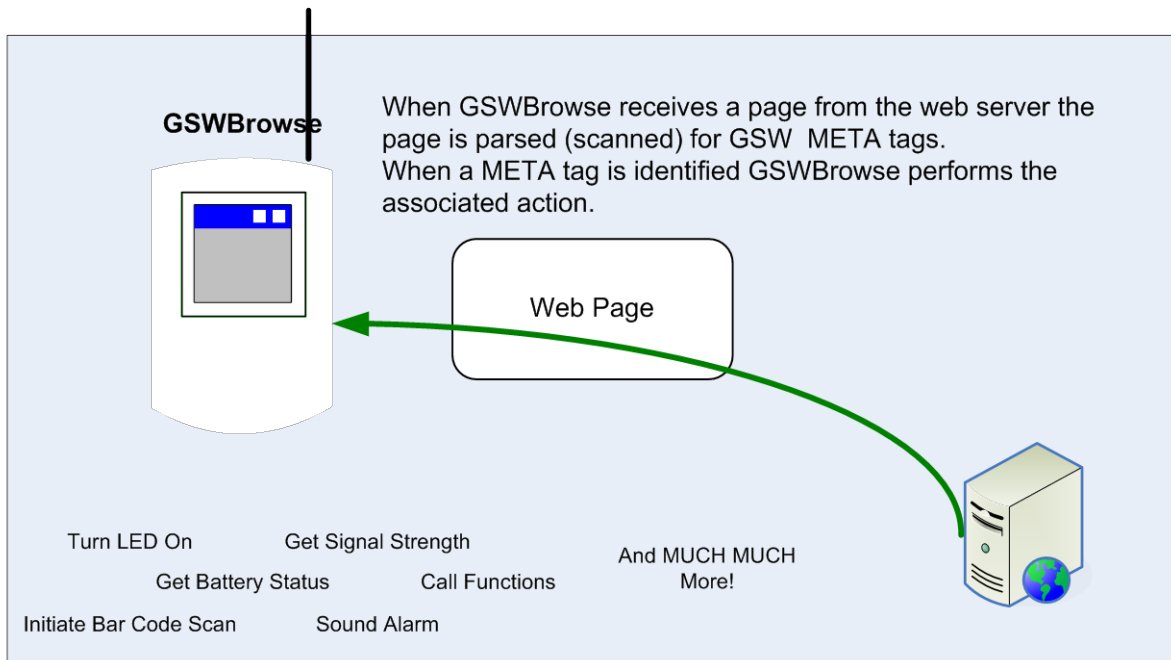


Figure 33: Meta Tag Processing Overview

SYNTAX

```
<META HTTP-EQUIV="name" CONTENT="content">
```

Where

Name is the command part of GSWAC⁶. and

Content The parameter part of GSWAC. Typically a value or a JavaScript function or a URL.

⁶ See Figure 32: GSW Commands Action Code Definition Diagram on page 30

Meta Tag Syntax

Meta Tag syntax is static and is inside the header of the document. The associated action is initiated when the document loading is complete. Meta Tags with the HTTP-EQUIV attribute can be used to invoke GSWBrowse codes⁷.

For example, to set the time to 19:20 you could use the Meta Tag `gswbrowse_SetTime` as shown below.

```
<META HTTP-EQUIV="gswbrowse_SetTime" Content="19:20">
```

Figure 34: Meta Tag Syntax Example

Once the page is loaded, (with the above Meta Tag specified) the time on the device will be set to **19:20**.

Multiple GSWBrowse Meta Tags can be used in the web page header section and they are processed in the order which they appear.

Navigational Syntax

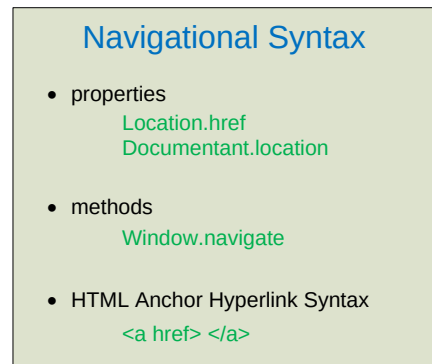


Figure 35: Navigational Syntax Properties, Methods and HTML

Navigational syntax is dynamic and is inside the body of the document. It is initiated based on events such as buttons, functions, etc. as documented in JavaScript and HTML manuals. Navigation Syntax is programming that invokes hyperlinks. GSWBrowse uses Pseudo-URLs to invoke GSWBrowse Action Codes by intercepting navigation requests.

With navigational syntax Georgia SoftWork's Action Codes are identified by GSWBrowse by using a common and familiar technique called **Pseudo-URL Prefix**. The Pseudo URL Prefix identifies the scheme used to determine the target of the navigation.

⁷ Familiarity with the HTML HTTP-EQUIV attribute is helpful.

To specify the GSWBrowse URL Prefix scheme, the Georgia SoftWorks Action Command must be prefixed with the characters **gswac** followed by a colon - "**gswac:**". The component parts of the Pseudo URL are shown below.

<u>GSW Action Code Scheme</u>	<u>Command</u>	<u>Parameter</u>
gswac:	gswbrowse_SetTime	19:20

Figure 36: Georgia SoftWorks Navigation Syntax Parts

The Pseudo-URL syntax conforms to the syntax of the property or method of the navigation technique used.

Anchor Tag

The diagram below shows an example of the **Anchor Tag** with the HREF attribute used to invoke commands.

```
<a href="gswac:gswbrowse_SetTime,19:20">Set Time to 19:20</a>
```

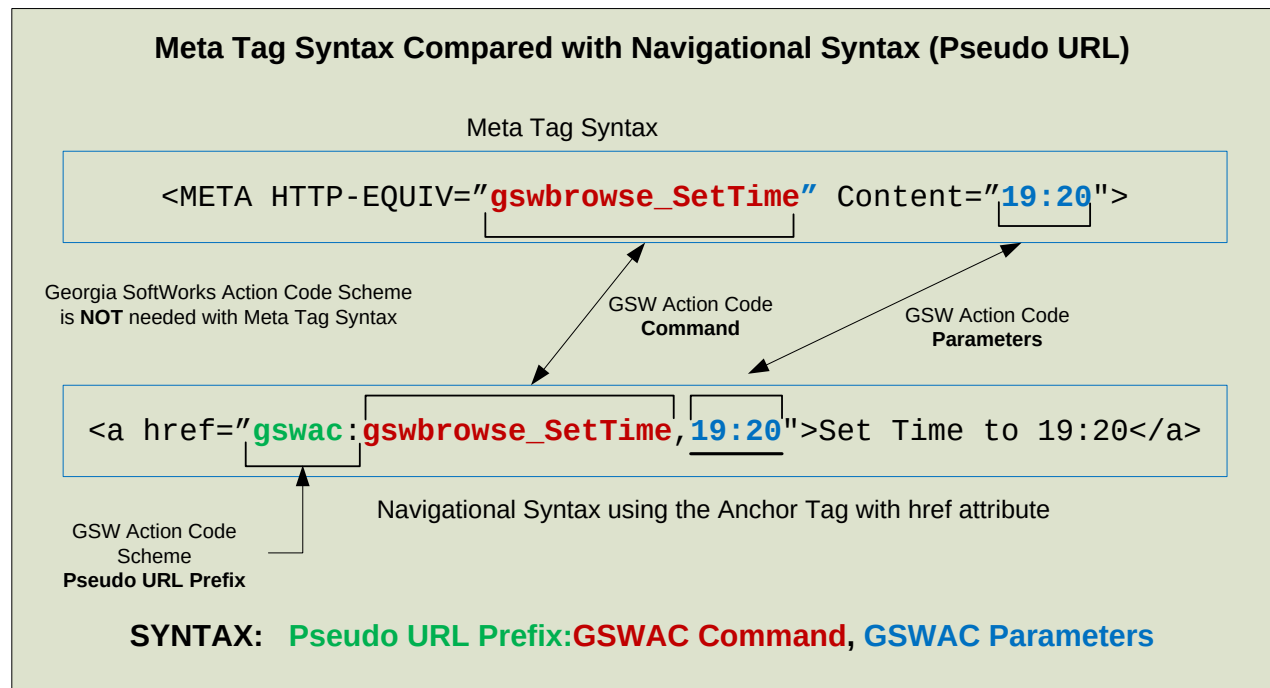


Figure 37: Meta Tag Syntax Compared with Navigational Syntax

The Anchor Tag uses the same GSWAC as the Meta Tag but is formatted for Navigation using the Pseudo-URL syntax. The action will be performed when the link is clicked.

HREF Property

The **href property** of the location and document objects can be used to invoke commands from JavaScript.

```
location.href = "gswac:gswbrowse_SetTime,19:20">
```

```
document.href = "gswac:gswbrowse_SetTime,19:20">
```

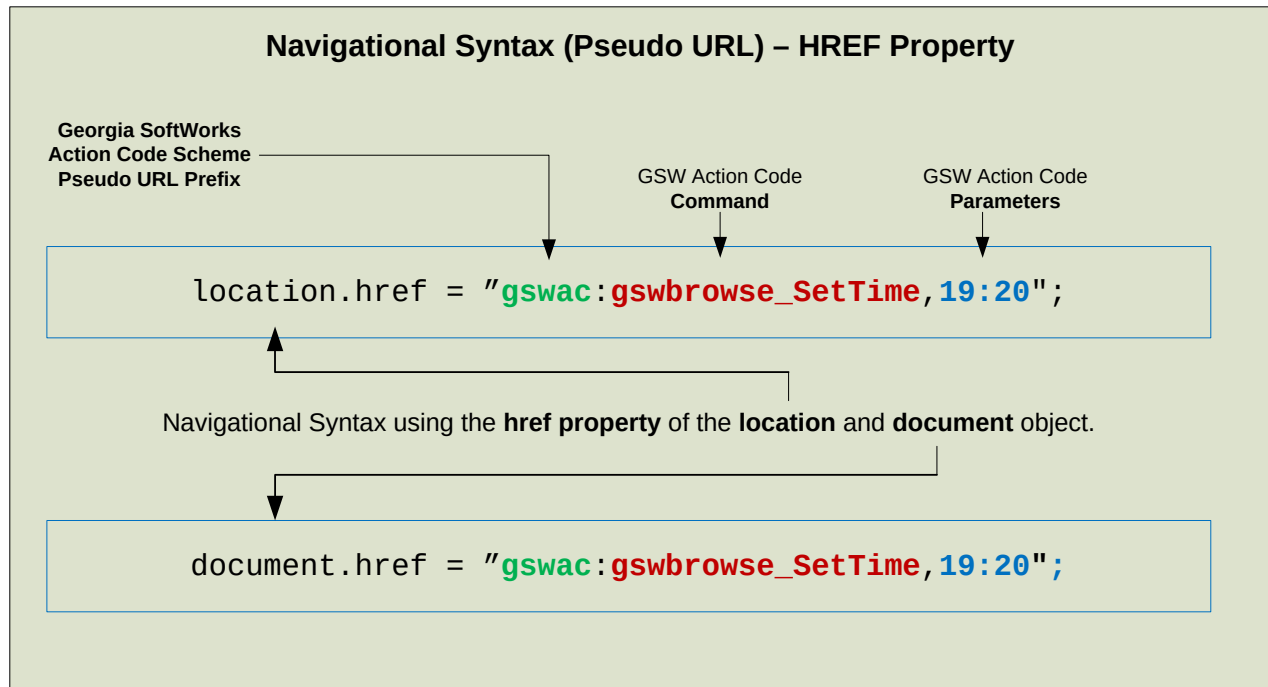


Figure 38: Navigation Syntax - HREF Property



NAVIGATE Method

The **Navigate method** of the Window object can be used to invoke commands.

```
window.navigate("gswac:gswbrowse_SetTime,19:20")
```

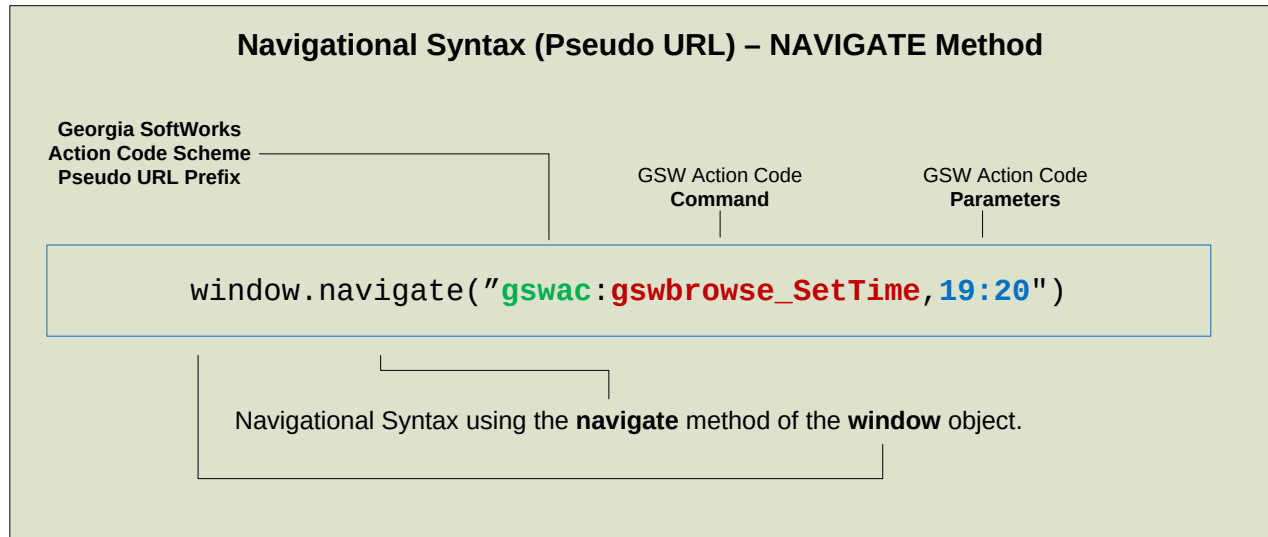


Figure 39: Navigation Syntax - NAVIGATE Method

Window External Syntax

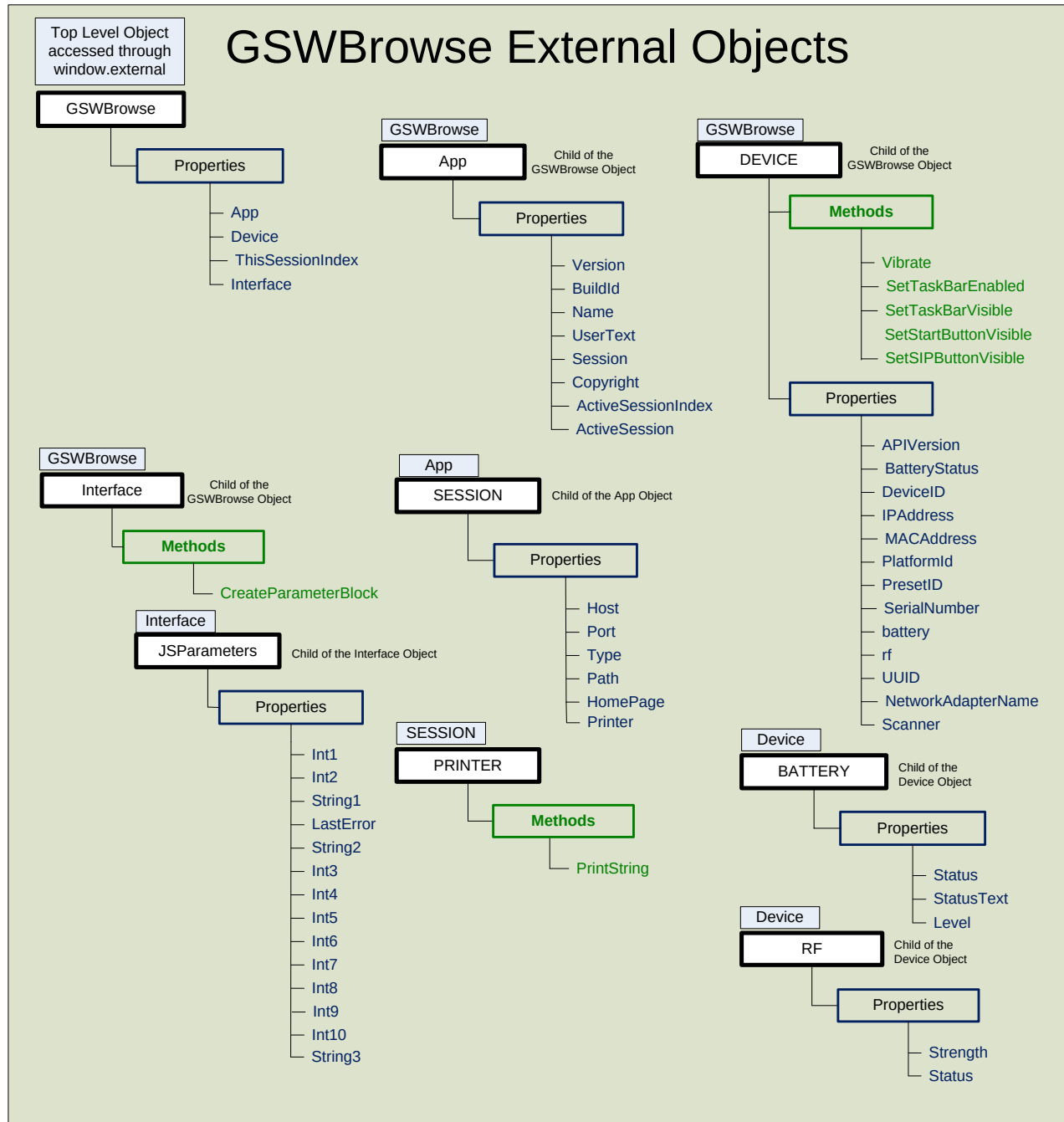


Figure 40: GSWBrowse Programmable External Objects

GSWBrowse is the top level object and is accessed as window.external.



Additional vendor dependant functionality is available via the scripting model of GSWBrowse through the window.external extensions. Windows External Syntax is used to access device, application and operating system features.

Some features are available only thru window.external functionality.

Send Menu Commands Overview

GSWBrowse offers programmatic access to the GSWBrowse menus. Any task that can be performed from the GSWBrowse menus can also be performed programmatically.



Developers Quick Reference

	<u>Hooking the Keyboard</u>	
	gswbrowse_onkey	Intercept a specific key and define action
	gswbrowse_onallkeys	Intercept all keys and define action to take
	gswbrowse_homekey	Enables/Disable F5 as the Home Key. When pressed navigate to the configured HOME URL
	Battery	
	gswbrowse_batterynavigate	Periodically obtain battery status information
	gswbrowse_batterytimerinterval	Specifies the interval to check battery useful
	window.external.device.batterystatus	Obtain the charged state of the battery
	window.external.device.battery.statustext	Text description of the battery charge state
	window.external.device.battery.status	Numeric value for the charged state of the battery
	window.external.device.battery.level	Charged state of the battery as a percentage
	Resume Device Operation after Suspension	
	gswbrowse_poweron	Specifies action to initiate when the device resumes power after suspension
	Timer-Based Actions	
	gswbrowse_timernavigate	Launches a JavaScript Function or URL at specified intervals
	gswbrowse_timerinterval	Specifies the interval to launch gswbrowse_timernavigate
	Browser Navigation Error Handling	
	gswbrowse_errornavigate	Launches a JavaScript function or URL when a navigation error is detected by the browser
	Exit from Browser	
	gswbrowse_command	Causes GSWBrowse to Exit
	gswbrowse_application	Causes GSWBrowse to Exit
	Controlling Input Panel (soft keyboard)	
	gswbrowse_sip	Hide/Display Soft Input Panel (SIP)
	gswbrowse_sipup	Displays the Soft Input Panel(SIP)
	gswbrowse_movesip	Moves the Soft Input Panel (SIP) to a specified location using x and y coordinates



	Date and Time	
	gswbrowse_settime	Set the System Time on the device
	gswbrowse_setdate	Set the System Date on the device
	Font Size	
	gswbrowse_fontsize	Change the Font Size relative to the size specified in the HTML content
	Cursor Position	
	gswbrowse_curpos	Move the cursor to a specified location using x and y coordinates
	Warm Boot and Soft Boot	
	gswbrowse_reboot	Forces the device to reboot (warm or cold)
	RF Signal Strength	
	gswbrowse_signalnavigate	Obtain WLAN Signal Information
	window.external.device.rf.strength	Obtain the strength of the RF Signal
	window.external.device.rf.Status	Numeric value that represents the status of the RF Signal
	Scanner Operation	
	gswbrowse_scannernavigate	Intercept result of a scan
	GSWBrowse Application Control	
	window.external.ThisSessionIndex	Obtain the session number of your execution context
	window.external.app.Session	Obtain session object corresponding to the provided index of the browser instance
	window.external.app.ActiveSessionIndex	Get/Set the Visible Browser Index
	window.external.app.session(idx).host	Obtain host part of URL for specified session
	window.external.app.session(idx).port	Obtain port number for specified session
	window.external.app.session(idx).type	Obtain session type for specified session
	window.external.app.session(idx).path	Obtain path for specified session
	window.external.app.session(idx).homepage	Get/Set home page URL for specified session
	Printing (<i>BlueTooth, IRDA, Network, Serial</i>)	
	gswbrowse_ZebraLabel_Print	
	gswbrowse_PLSeriesLabel_Print (same as gswbrowse_ZebraLabel_Print)	Supported for backward compatibility
	gswbrowse_ZebraLabel_Complete	Send Output to the Printer
	window.external.app.session(idx).Printer.PrinterString	Send Output to Printer



GSWBrowse about Information	
window.external.app.Version	Obtain GSWBrowse release version number
window.external.app.BuildId	Obtain GSWBrowse Build Id which is a unique identifier for each software build
window.external.app.Name	Name of the current application running
window.external.app.Copyright	Obtain copyright text for GSWBrowse
Persistent Storage	
window.external.app.UserText	Indexed property that stores user data strings in persistent storage
Device Information	
window.external.device.DeviceID	Read the Device Identification string
window.external.device.IPAddress	Read the Device IP Address
window.external.device.MACAddress	Read the Device Media Access Control (MAC) Address
window.external.device.PlatformId	Read the Windows CE Platform Id
window.external.device.PresetId	Read the Windows CE Preset Id
window.external.device.SerialNumber	Read the GSW Product Id which is used for software registration
window.external.device.UUID	Read the Universally Unique Identifier (UUID) of the device
Network Adapter Name	
window.external.device.NetworkAdapterName	Read text name for the network adapter
Vibrating the Device (LXE Only)	
window.external.device.Vibrate	Cause the device to Vibrate for a specified duration of time
Controlling the Taskbar	
window.external.device.SetTaskBarEnabled	Enable/Disable the taskbar
window.external.device.SetTaskBarVisible	Hide/Show the taskbar
window.external.device.SetStartButtonVisible	Hide/Show the Start Button
window.external.device.SetSIPButtonVisible	Hide/Show the Soft Input Panel (SIP) Button
Vendor API Version (LXE Only)	
window.external.device.APIVersion	Read the Version Number of the Vendor's Application Programming Interface (API)



LXE SCANNER APIs window.external. device.scanner.lxescanner	
For all LXE SCANNER API's please consult the appropriate LXE manual for detailed and current information. These API's work as pass-throughs to LXE APIs provided in lxapi.dll ⁸	
<u>.LXEHasIntScanner</u>	Determine if device has an internal scanner
<u>.LXEHasIntRFID</u>	Determine if device has internal RFID Reader
<u>.LXEScannerDisable</u>	Cause scanner driver to disable operation
<u>.LXEScannerEnable</u>	Return scanner driver to normal operation
<u>.LXEScannerInputDisable</u>	Disable input (scan keys & trigger button)
<u>.LXEScannerInputEnable</u>	Enable input after disabled
<u>.LXEScannerReset</u>	Cause scanner driver to reload registry settings
<u>.LXEScannerForcePower</u>	Forces serial port to supply power to scanner
<u>.LXEScannerForceScan</u>	Force Scan
<u>.LXEScannerStart</u>	Start scan operation on internal scanner
<u>.LXEScannerStop</u>	Terminate scan operation on internal scanner
<u>.LXEScannerGetStatus</u>	Returns the status of the scanner
<u>.LXEScannerGetData</u>	Obtain the scanned string
<u>.LXEScannerPutData</u>	See LXE documentation
<u>.LXEScannerPowerON</u>	Apply power to all active scanner devices
<u>.LXEScannerPowerOff</u>	Power down all active scanner devices
<u>.LXEScannerKeysOff</u>	Stop processing scanned data as keystroke msgs
<u>.LXEScannerKeysOn</u>	Start processing scanned data as keystroke msgs
<u>.LXEScannerKeyStatus</u>	Determine if scanner wedge is buffering data or processing as keystroke messages
<u>.LXEScannerAttachPort</u>	Changes wedge port attached to the scanner driver as specified
<u>.LXEScannerSetKey</u>	Changes the action of the scan key specified
<u>.LXEScannerSetSerial</u>	Changes the configuration of scanner the serial port
<u>.LXEScannerSetPower</u>	Changes the value of the scanner serial port power pin and saves to registry
<u>.LXEScannerStripLead</u>	Enable/Disable stripping of characters from beginning of barcode data
<u>.LXEScannerStripTrail</u>	Enable/Disable stripping of characters from end of barcode data
<u>.LXEScannerCtrlCodeOff</u>	Disables processing of ASCII values less than 0x20 when data is processed as keystroke messages
<u>.LXEScannerCtrlCodeOn</u>	Enables processing of ASCII values less than 0x20 when data is processed as keystroke messages
<u>.LXEScannerPrefix</u>	Enable/Disable adding prefix to beginning of barcode data
<u>.LXEScannerSuffix</u>	Enable/Disable adding Suffix to end of barcode data
<u>.LXEScannerOption</u>	Enable/Disable symbology option
<u>.LXEScannerIsOptionEnabled</u>	Determine if given symbology is enabled/disabled
<u>.LXEScannerGetParam</u>	Get the value of a hardware specific parameter directly from the scanner hardware

⁸ Obtain the file from LXE



LXE SCANNER APIs - Continued window.external.device.scanner.lxscanner	
<u>.LXEScannerSetParam</u>	Set value of a hardware specific parameter
<u>.LXEScannerSymStripLead</u>	Enable/Disable stripping of characters from the beginning of barcode data for specified symbology
<u>.LXEScannerSymStripTrail</u>	Enable/Disable stripping of characters from the end of barcode data for specified symbology
<u>.LXEScannerSymPrefix</u>	Enable/Disable adding prefix to beginning of barcode data for specified symbology
<u>.LXEScannerSymSuffix</u>	Enable/Disable adding suffix the end of barcode data for specified symbology
<u>.LXEScannerSymEnable</u>	Enable/Disable specified symbology
<u>.LXEScannerSymIsEnabled</u>	Determine the current enable state of the parsing engine for the specified symbology
<u>.LXEScannerSymGetConfig</u>	Gets all parameters for specified symbology
<u>.LXEScannerSymSetConfig</u>	Sets all parameters for a symbology
<u>.LXEScannerSymRemove</u>	Removes all parameters for specified symbology by deleting the registry key
GSWBrowse Interface Helper Object	
<u>window.external.interface.CreateParameterBlock()</u>	Create a Helper Object for parameter passing

Developers Detailed Reference

The following section provides the detailed reference on the Georgia SoftWorks Action Codes and external properties and methods.

For each Georgia SoftWorks Action Code an information box is located at the top right of the page that provides quick information with respect to the **syntax** and **scope**.

Scope / Syntax

The syntax indicates the forms that this functionality can be invoked. Multiple syntaxes may be valid for a particular function. The options are:

- Meta Tag
- Navigational
- Windows External

The example below indicates that the functionality is available using Meta Tag and Navigational Syntax.

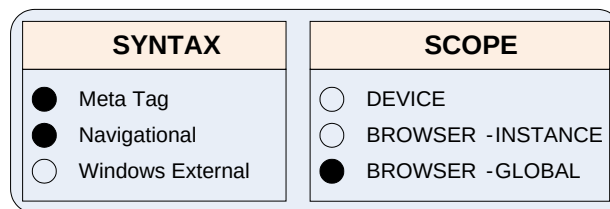


Figure 41: Syntax / Scope Diagram

Additionally the scope of the functionality is also important. The scope identifies the range of impact of the command. GSWBrowse can have five (5) browsers. Some commands only impact the specific browser instance and others may impact all browsers.

Likewise some functionality may only impact the device. The options for scope are:

- Device
- Browser - Instance
- Browser – Global

The example above indicates that the functionality impacts all browsers.

Developer Attention Boxes

Other important information is conveyed using developer attention boxes.



Developer attention boxes are located on the page when a *special or particularly useful* aspect of this functionality should be noticed. Most of the developer attention boxes are self explanatory however a few examples follow:

Examples:

- GSW ONLY

GSW Only – At the time of this writing, GSW is the only vendor that provides this feature.



- LXE Devices ONLY. This External is Specific to LXE Devices

LXE Only –This feature only applies to LXE devices.



- READ ONLY. Any attempt to set this property will result in a runtime error.

This is a READ ONLY property.

Each GSWAC description defines the important aspects of the command. The command, parameters, syntaxes and examples are shown.

For parameters the pipe “|” symbol means OR. For example if a command has the parameter listed as:

JavaScript Function | URL

This means that either a JavaScript Function OR a URL can be a valid parameter.

Often the entire content is shown for a web page in order to provide the context for examples. When these examples are shown, the text significant to the GSWAC being described is usually in the color **red** to help it stand out (for those viewing online or a color printed copy).



Hooking the Keyboard Detail

Category: Hooking the Keyboard					
GSWBROWSE_ONKEY	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☒ Meta Tag ☒ Navigational ☐ Windows External </td><td> ☐ DEVICE ☒ BROWSER - INSTANCE ☐ BROWSER - GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER - INSTANCE ☐ BROWSER - GLOBAL
SYNTAX	SCOPE				
☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER - INSTANCE ☐ BROWSER - GLOBAL				

gswbrowse_ONKEY

The GSWBROWSE_ONKEY tag changes the action of a specific key by assigning a JavaScript function or a URL to the specific (hardware or soft keyboard) key on the device.

Command: gswbrowse_OnKey[xyz]

Parameter(Content): JavaScript Function | URL

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_OnKey[xyz]" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_OnKey[xyz],content";
document.location = "gswac:gswbrowse_OnKey[xyz],content";
```

Methods

```
windows.navigate("gswac:gswbrowse_OnKey[xyz],content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_OnKey[xyz],content">
```

Where

xyz is the key mapped and can be either be
a hexadecimal value preceded with 0x
Example: gswbrowse_OnKey0x56
or up to a three digit decimal value
Example: gswbrowse_OnKey086
or a Virtual Key Name
Example: gswbrowse_OnKey_NUMPAD8
or gswbrowse_OnKeyVK_NUMPAD8

The list of virtual key names can be viewed on [page 241](#)

Content is a JavaScript function

Example: Javascript:OnNUMPAD8();

OR a URL

Example: http://acme_server/page_a.htm



Example: GSWBROWSE_ONKEY

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_OnKey0x56" CONTENT="Javascript:OnV();">
<META HTTP-EQUIV="gswbrowse_OnKeyVK_A" CONTENT="http://acme_server/page_a.htm">
<META HTTP-EQUIV="gswbrowse_OnKey_NUMPAD8" CONTENT="Javascript:OnNUMPAD8();">
<META HTTP-EQUIV="gswbrowse_OnKey48" CONTENT="Javascript:On48();">

<script>
function OnV()
{
    Alert(" V was pressed");
}
function OnNUMPAD8()
{
    Alert(" 8 on the number pad was pressed");

}

function On48()
{
    Alert(" Digit 0 was pressed");

}
</script>
</head>
<body>
<p>Pressing 'a' will cause navigation to 'page_a.htm'</p>
<p>The V, the number 0 and the number pad 8 will each display an alert when
pressed.</p>
</body>
</html>
```



Category: Hooking the Keyboard

GSWBROWSE_ONALLKEYS**SYNTAX**

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER - INSTANCE
- ☐ BROWSER - GLOBAL

gswbrowse_ONALLKEYS

The GSWBROWSE_ONALLKEYS command launches a JavaScript function or a URL when any key is pressed on the device. The specified function must have one argument which is the virtual keycode pressed. This is a true key interception; it takes place regardless of where the focus is in GSWBrowse, which is superior to native DHTML functionality

Command: gswbrowse_onallkeys

Parameter(Content): JavaScript Function | URL

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_onallkeys" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_onallkeys,content";
document.location = "gswac:gswbrowse_onallkeys,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_onallkeys,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_onallkeys,content">
```

Where

Content is a JavaScript function

Example: Javascript: *onallkeys*(%s);

OR a URL

Example: http://acme_server/valicate_order.htm?vk=%s

In both cases the %s is replaced with the decimal value of virtual keycode of the key pressed.



- This is a TRUE KEY INTERCEPTION.
- It takes place regardless of where the focus in GSWBrowse which is superior to native DHTML functionality.

Example: GSWBROWSE_ONALLKEYS

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_OnAllKeys" CONTENT="Javascript:OnAllVK(%s);">
<script>
function OnAllVK(vk)
{
    Alert( vk + " was pressed");
}
</script>
</head>
<body>
<p>Pressing any key will cause an alert when pressed.</p>
</body>
</html>
```



Category: Hooking the Keyboard

GSWBROWSE_HOMEKEY**SYNTAX**

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER - INSTANCE
- ☐ BROWSER - GLOBAL

gswbrowse_HOMEKEY

The GSWBROWSE_HOMEKEY command enables Function Key F5 as the HomeKey. When pressed, the HOMEKEY navigates to the configured HOME URL. Default – Disabled

Command: gswbrowse_HomeKey

Parameter(Content): Enabled|Disabled

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_HomeKey" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_onallkeys,content";  
document.location = "gswac:gswbrowse_onallkeys,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_onallkeys,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_onallkeys,content">
```

Where

Example: Meta Tag Syntax

```
<META HTTP-EQUIV="gswbrowse_HomeKey" CONTENT="Enabled">
```

Example: Navigation Syntax

```
document.location = "gswac:gswbrowse_homekey,"Disabled";
```



Battery Information

Category: Battery Information

GSWBROWSE_BATTERYNAVIGATE

SYNTAX

- ☒ Meta Tag
- ☒ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

gswbrowse_BATTERYNAVIGATE

The GSWBROWSE_BATTERYNAVIGATE command launches a JavaScript Function or URL at specified intervals passing strings that contain the battery status information. The **time interval must** be specified with the GSWBROWSE_BATTERTIMERINTERVAL command for this command to work.

Command: gswbrowse_BatteryNavigate

Parameter(content): JavaScript Function | URL

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_BatteryNavigate" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_batterynavigate,content";
document.location = "gswac:gswbrowse_batterynavigate,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_batterynavigate,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_batterynavigate,content">
```



- Timer Interval Must Be Specified when using Meta Tag or Navigation Syntax
- gswbrowse_BatteryNavigate does not execute until the BatteryTimer Interval expires
- Reloading the page cancels gswbrowse_BatteryNavigate and gswbrowse_BatteryTimerInterval
- Navigation away from the page cancels gswbrowse_BatteryNavigate and gswbrowse_BatteryTimerInterval

Where

Content is a JavaScript function

Example: `Javascript:BatteryTest( '%s', '%s', '%s', '%s' );`

OR a URL

Example:

`http://acme_server/battstat?AC=%s&strength=%s&backup=%s&chemistry=%s`

Return

The Battery status information passed to the JavaScript Function or URL is:

AC Line Status

Possible Values

“Offline”

“Online”

“Unknown”

Battery Life Percent

A number in the range of 0 – 100, or 255 if unknown

Battery Backup Life Percent

A number in the range of 0 – 100, or 255 if unknown

Battery Chemistry

Possible Values

“BATTERY_CHEMISTRY_ALKALINE”

“BATTERY_CHEMISTRY_NICD”

“BATTERY_CHEMISTRY_NMH”

“BATTERY_CHEMISTRY_LION”

“BATTERY_CHEMISTRY_LIPOLY”

“BATTERY_CHEMISTRY_ZINCAIR”

“BATTERY_CHEMISTRY_UNKNOWN”

JavaScript Pseudo-URL Target

When the Meta Tag is processed the JavaScript function will be called with the battery status values passed to the JavaScript function in the order as described in Figure 42 .

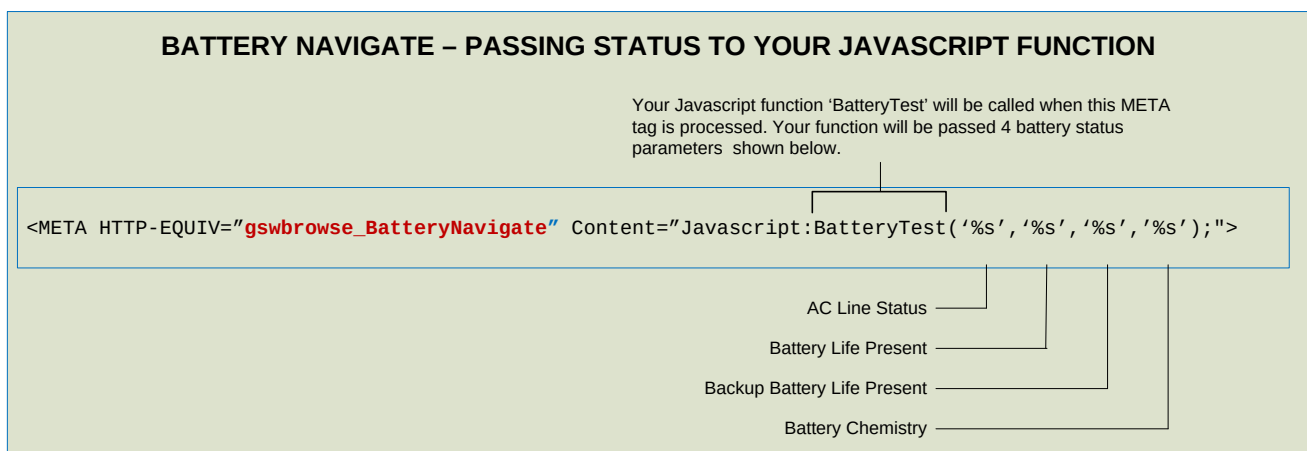


Figure 42: Passing Battery Status to Your JavaScript Function

URL Target

If a URL is specified then it is parsed and the pattern replacement described below will occur before the actual navigation takes place. Each pattern specifies the name of the of the battery information of interest. The exact syntax must be used. See Figure 43 for more detail.

Using a URL target with this Meta Tag allows passing values from the client to the server as if a form was just submitted. The values are processed on the server using common value retrieval techniques such as CGI, ASP, etc. The format of the URL must adhere to the standard HTTP GET request with Form parameters.

If **'AC=%s'** is found then %S will be replaced with the AC line values as specified above.

If **'strength=%s'** is found then %S will be replaced with the Battery Life Percent as specified above.

If **'backup=%s'** is found then %S will be replaced with the Backup Battery Life Percent as specified above.

If **'chemistry=%s'** is found then %s will be replaced with the Battery Chemistry as specified above.

The URL target has the capability to just retrieve the status of a select number of the statuses.

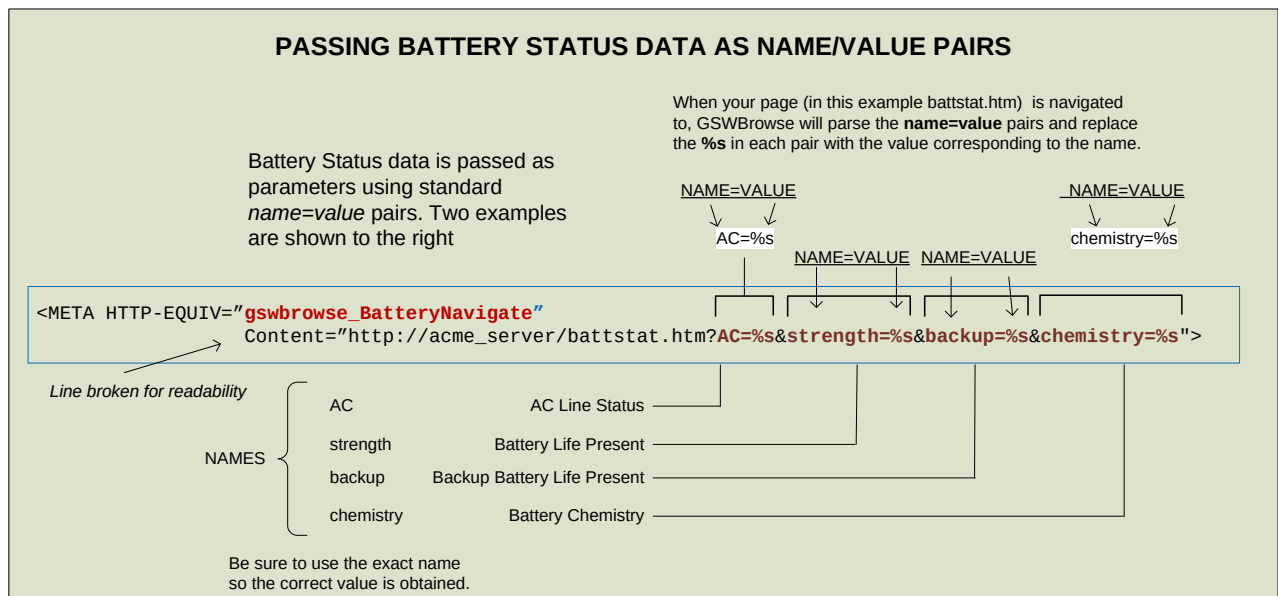


Figure 43: Battery Navigate - Passing Form Data as NAME/VALUE pairs



Category: Battery Information

GSWBROWSE_BATTERYTIMERINTERVAL**SYNTAX**

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

gswbrowse_BATTERYTIMERINTERVAL

The GSWBROWSE_BATTERYTIMERINTERVAL command specifies the time (in milliseconds) interval that gswbrowse_BatteryNavigate command is launched.

Command: gswbrowse_BatteryTimerInterval

Parameter (content): Decimal number between 0 and 4294967275
Units: milliseconds

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_BatteryTimerInterval" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_BatteryTimerInterval,content";  
document.location = "gswac:gswbrowse_BatteryTimerInterval,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_BatteryTimerInterval,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_BatteryTimerInterval,content">
```

Where

Content is the decimal time in milliseconds with a range between 0 and 4294967295. Comma's are not allowed in the number.



- Time is in decimal milliseconds. (1/1000 of a second)
- Reloading the page cancels gswbrowse_BatteryNavigate and gswbrowse_BatteryTimerInterval
- Navigation away from the page cancels gswbrowse_BatteryNavigate and gswbrowse_BatteryTimerInterval



Example: GSWBROWSE_BATTERYTIMERINTERVAL

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_BatteryNavigate"
CONTENT="Javascript:BatteryTest('%s','%s','%s','%s');">
<META HTTP-EQUIV="gswbrowse_BatteryTimerInterval" CONTENT="2000">
<META HTTP-EQUIV="gswbrowse_TimerNavigate" CONTENT="Javascript:TimerTest('%s');">
<META HTTP-EQUIV="gswbrowse_TimerInterval" CONTENT="1000">

<script language=javascript>
function BatteryTest(a,b,c,d)
{
  document.getElementById("batterydiv").innerHTML= a + " " + b + " " + c + " " +d;
}
function TimerTest(a)
{
  document.getElementById("battery_statustext").innerHTML=
window.external.device.battery.statustext;
  document.getElementById("battery_status").innerHTML=
window.external.device.battery.status;
  document.getElementById("battery_level").innerHTML=
window.external.device.battery.level;
}
</script>
</head>
<body>
<p>The BatteryNavigate tag assumes your device has a battery. No
navigation or script call shall occur if a battery is not found.</p>
Current battery state: <div id="batterydiv">junk</div>
<p>External battery properties</p>
battery.statustext: <div id="battery_statustext">junk</div>
battery.status: <div id="battery_status">junk</div>
battery.level: <div id="battery_level">junk</div>
</body>
</html>
```



Category: Battery Information

BATTERYSTATUS**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.batterystatus`

The **batterystatus** is a property of the **device** object. It provides the charged state of the battery.

Property: `batterystatus`

Data Type: String

☒ Read ☐ Write

Return Possible Return Values are:

unknown
no_battery
charging
critical
low
medium
high

EXTERNAL SYNTAX`window.external.device.batterystatus`**IDL DEFINITION**

```
[propget, id(1)] HRESULT BatteryStatus([out, retval] BSTR *pVal);
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Example: BATTERY STATUS

```
<html>
<head>

<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_BatteryNavigate"
CONTENT="Javascript:BatteryTestfun('%s','%s','%s','%s');">
<META HTTP-EQUIV="gswbrowse_BatteryTimerInterval" CONTENT="2000">
<META HTTP-EQUIV="gswbrowse_TimerNavigate" CONTENT="Javascript:TimerTestfun('%s');">
<META HTTP-EQUIV="gswbrowse_TimerInterval" CONTENT="1000">

<script language=javascript>
function BatteryTestfun(a,b,c,d)
{
    document.getElementById("batterydiv").innerHTML= a + " " + b + " " + c + " " +d;
}
function TimerTestfun(a)
{
    document.getElementById("device_batterystatus").innerHTML=
window.external.device.device.batterystatus;
    document.getElementById("battery_statustext").innerHTML=
window.external.device.battery.statustext;
    document.getElementById("battery_status").innerHTML= window.external.device.battery.status;
    document.getElementById("battery_level").innerHTML= window.external.device.battery.level;
}
</script>

</head>

<body>
<p>The BatteryNavigate tag assumes your device has a battery. No
navigation or script call shall occur if a battery is not found.</p>
Current battery state: <div id="batterydiv">junk</div>
<p>External battery properties</p>
device.batterystatus: <div id="device_batterystatus">junk</div>
battery.statustext: <div id="battery_statustext">junk</div>
battery.status: <div id="battery_status">junk</div>
battery.level: <div id="battery_level">junk</div>
</body>
</html>
```



Category: Battery Information

STATUS**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.battery.status

Status is a property of the battery object. It provides a numeric value for the charged state of the battery.

Property: status

Data Type: Number

☒ Read ☐ Write

Return

The Return value is a number representing the battery charge status. The status is cumulative. For example charging(8) and low(2) may be a combined status with a resulting value of 10. Use bit testing for examination of the status.

255 means unknown
128 means no_battery
8 means charging
4 means critical
2 means low
1 means high

EXTERNAL SYNTAX

window.external.device.**battery.status**

IDL DEFINITION

```
[propget, id(2)] HRESULT Status([out, retval] long *pVal);
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Category: Battery Information

STATUSTEXT**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.battery.statustext`

StatusText is a property of the **battery** object. It provides a text description of the battery charge state. Read Only.

Property: statustext

Data Type: String

☒ Read ☐ Write

Return

The Return value is a string describing the battery charge status. Note that the returned string is not cumulative as with the `batterystatus`. The operating system determines which information is most important.

Unknown
No_battery
Charging
Critical
Low
Medium
High

EXTERNAL SYNTAX`window.external.device.battery.statustext`**IDL DEFINITION**

```
[propget, id(1), helpstring("property StatusText")]  
HRESULT StatusText([out, retval] BSTR *pVal);
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Category: Battery Information

LEVEL**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.battery.level

Level is a property of the **battery** object. It provides the charged state of the battery as a percentage.

Property: level

Data Type: Number

☒ Read ☐ Write

Return The return value is a number representing the percent of remaining battery charge.

0 - 100 is the Percent of Battery Charge
255 means Unknown

EXTERNAL SYNTAX

window.external.device.**battery.level**

IDL DEFINITION

```
[propget, id(3)] HRESULT Level([out, retval] long *pVal);
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



RF Signal Strength

Category: RF Signal

GSWBROWSE_SIGNALNAVIGATE

SYNTAX

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

gswbrowse_SIGNALNAVIGATE

The GSWBROWSE_SIGNALNAVIGATE command launches a JavaScript Function or URL passing strings that contain the WLAN Signal Information.

Command: gswbrowse_SignalNavigate

Parameter(content): JavaScript Function | URL

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_SignalNavigate" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_SignalNavigate,content";
document.location = "gswac:gswbrowse_SignalNavigate,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_SignalNaviagte,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_SignalNavigate,content">
```

Where

Content is a JavaScript Pseudo-URL function
Example: `Javascript:OnSignal('%s','%s','%s');`

OR a URL
Example: `http://acme_server/signal.htm?strength=%s&ssid=%s&mac=%s`



- Although similar in name, the GSWBROWSE_SIGNALNAVIGATE is not a Timer based command.
- Unlike the gswbrowse_BatteryNavigate and gswbrowse_TimerNavigate, the gswbrowse_SIGNALNAVIGATE does not have a delay associated with a timer. It is executed after the page is loaded or when navigation occurs.

Return

The Signal status information passed to the JavaScript Function or URL is:

Signal Strength A number in the range of 0 – 100, or 255 if unknown

**Extended Service Set ID (ESSID or SSID)**

The name of the Wireless Network,

MAC Address

Media Access Control address of the device

JavaScript Pseudo-URL Target

When the Meta Tag is processed the JavaScript function will be called with the WLAN status information. The status values passed to the JavaScript function are in the order as described below.

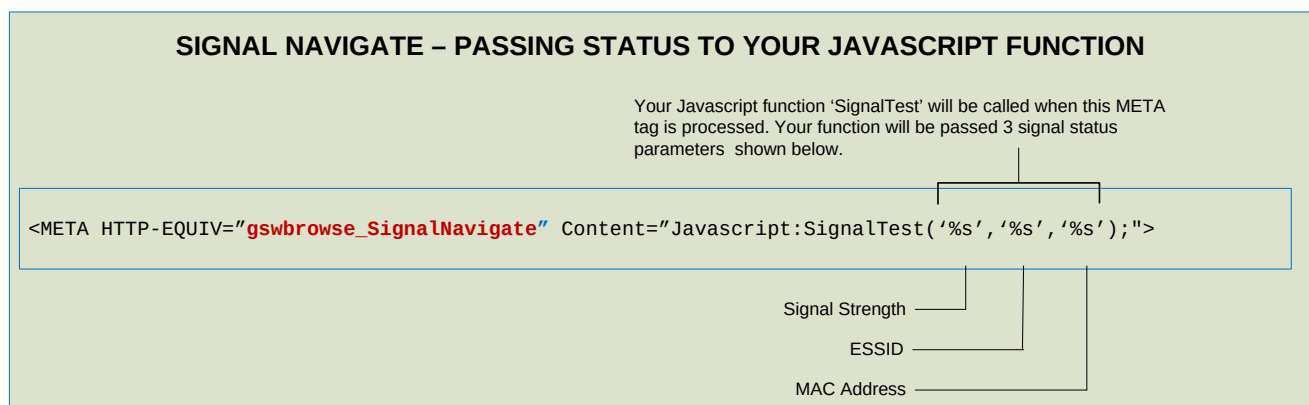


Figure 44: Passing Signal Status Information to Your JavaScript Function

URL Target

If a URL is specified then it is parsed and the pattern replacement described below will occur before the actual navigation takes place. Each pattern specifies the name of the of the signal information of interest. The exact syntax must be used. The parsing process is as follows:

If **'strength=%s'** is found then **%S** will be replaced with the signal strength value as specified above.

If **'essid=%s'** is found then **%S** will be replaced with the Extended Service Set Id (ESSID), commonly referred to as the network name or SSID.

If **'mac=%s'** is found then **%S** will be replaced with the Media Access Control(MAC) .

The URL target has the capability to retrieve a select number of the statuses.

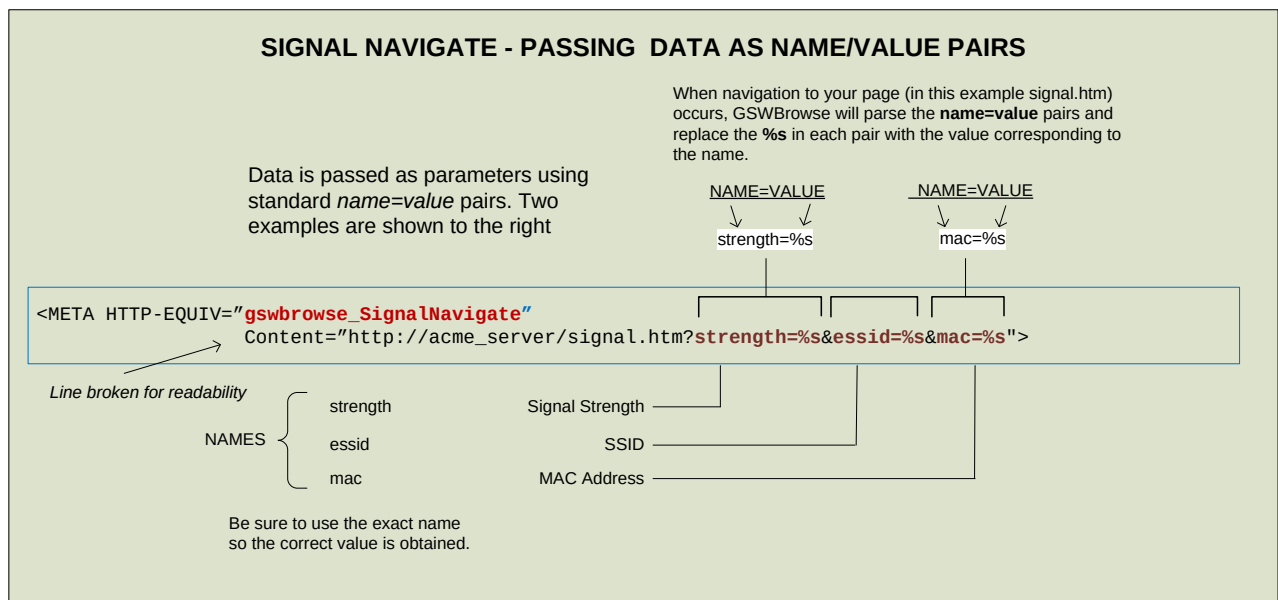


Figure 45: Signal Navigate - Passing Form Data as Name/Value Pairs.



Example: GSWBROWSE_SIGNALNAVIGATE

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="SignalNavigate" CONTENT="Javascript:OnSignal('%s', '%s', '%s');">
<META HTTP-EQUIV="TimerNavigate" CONTENT="Javascript:TimerTestfun('%s');">
<META HTTP-EQUIV="TimerInterval" CONTENT="1000">
<script id="script_OnSignal" type="text/javascript">
function OnSignal(s, ssid, mac)
{
    document.getElementById("ssiddiv").innerHTML=ssid;
    document.getElementById("macdiv").innerHTML=mac;
}
function TimerTestfun(a)
{
    document.getElementById("strengthdiv").innerHTML=
        window.external.device.rf.strength;
}
</script>
</head>

<body>
WLAN info:
<p>
Current strength: <div id="strengthdiv">junk</div>
Current SSID: <div id="ssiddiv">junk</div>
Current MAC: <div id="macdiv">junk</div>
</p>
</body>
```



Category: RF									
STRENGTH	<table border="1"><thead><tr><th>SYNTAX</th><th>SCOPE</th></tr></thead><tbody><tr><td>☐ Meta Tag</td><td>☒ DEVICE</td></tr><tr><td>☐ Navigational</td><td>☐ BROWSER -INSTANCE</td></tr><tr><td>☒ Windows External</td><td>☐ BROWSER -GLOBAL</td></tr></tbody></table>	SYNTAX	SCOPE	☐ Meta Tag	☒ DEVICE	☐ Navigational	☐ BROWSER -INSTANCE	☒ Windows External	☐ BROWSER -GLOBAL
SYNTAX	SCOPE								
☐ Meta Tag	☒ DEVICE								
☐ Navigational	☐ BROWSER -INSTANCE								
☒ Windows External	☐ BROWSER -GLOBAL								

window.external.device.rf.strength

Strength is a property of the **rf** object. It provides the strength of the RF Signal.

Property: strength

Data Type: NUMBER

☒ Read ☐ Write

Return The return value is a number representing the rf signal strength.

1 - 100 is the RF Signal Strength on a
 scale of 1 to 100
-1 means Unknown
-2 means Not Associated

EXTERNAL SYNTAX

window.external.device.**rf.strength**

IDL DEFINITION

```
[[propget, id(1), helpstring("property Strength")]  
HRESULT Strength([out, retval] long *pVal);
```



- READ ONLY. Any attempt to set this property will result in a runtime error.

Example: WINDOW.EXTERNAL.DEVICE.RF.STRENGTH

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="SignalNavigate" CONTENT="Javascript:OnSignal('%s', '%s', '%s');">
<META HTTP-EQUIV="TimerNavigate" CONTENT="Javascript:TimerTestfun('%s');">
<META HTTP-EQUIV="TimerInterval" CONTENT="1000">
<script id = "script_OnSignal" type="text/javascript">
function OnSignal(s, ssid, mac)
{
    document.getElementById("ssiddiv").innerHTML=ssid;
    document.getElementById("macdiv").innerHTML=mac;
}
function TimerTestfun(a)
{
    document.getElementById("strengthdiv").innerHTML=
        window.external.device.rf.strength;
}
</script>
</head>

<body>
WLAN info:
<p>
Current strength: <div id="strengthdiv">junk</div>
Current SSID: <div id="ssiddiv">junk</div>
Current MAC: <div id="macdiv">junk</div>
</p>
<p><a href="default.htm">Return to main menu</a></p>
</body>
```



Category: RF									
STATUS	<table border="1"><thead><tr><th>SYNTAX</th><th>SCOPE</th></tr></thead><tbody><tr><td>☐ Meta Tag</td><td>☒ DEVICE</td></tr><tr><td>☐ Navigational</td><td>☐ BROWSER -INSTANCE</td></tr><tr><td>☒ Windows External</td><td>☐ BROWSER -GLOBAL</td></tr></tbody></table>	SYNTAX	SCOPE	☐ Meta Tag	☒ DEVICE	☐ Navigational	☐ BROWSER -INSTANCE	☒ Windows External	☐ BROWSER -GLOBAL
SYNTAX	SCOPE								
☐ Meta Tag	☒ DEVICE								
☐ Navigational	☐ BROWSER -INSTANCE								
☒ Windows External	☐ BROWSER -GLOBAL								

window.external.device.rf.status

Status is a property of the **rf** object. It provides a number that represents the status of the RF Signal. Read Only.

Property: status

Data Type: Number

☒ Read ☐ Write

Return The Return value is a NUMBER which represents the RF Signal status

-1	means unknown
0	means unassociated
1	means poor
2	means fair
3	means good
4	means very good
5	means excellent

EXTERNAL SYNTAX

window.external.device.**rf.status**

IDL DEFINITION

```
[propget, id(2), helpstring("property Status")]  
HRESULT Status([out, retval] long *pVal);
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Example: WINDOW.EXTERNAL.DEVICE.RF.STATUS

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="SignalNavigate" CONTENT="Javascript:OnSignal('%s', '%s', '%s');">
<META HTTP-EQUIV="TimerNavigate" CONTENT="Javascript:TimerTestfun('%s');">
<META HTTP-EQUIV="TimerInterval" CONTENT="1000">
<script id="script_OnSignal" type="text/javascript">
function OnSignal(s, ssid, mac)
{
    document.getElementById("ssiddiv").innerHTML=ssid;
    document.getElementById("macdiv").innerHTML=mac;
}
function TimerTestfun(a)
{
    document.getElementById("strengthdiv").innerHTML=
        window.external.device.rf.strength;
document.getElementById("statusdiv").innerHTML=
        window.external.device.rf.status;

}
</script>
</head>

<body>
WLAN info:
<p>
Current strength: <div id="strengthdiv">junk</div>
Current SSID: <div id="ssiddiv">junk</div>
Current MAC: <div id="macdiv">junk</div>
Current STATUS: <div id="statusdiv">junk</div>

</p>
</body>
```



Resume After Suspension

Category: Battery Information									
GSWBROWSE_POWERON	<table border="1"> <thead> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> </thead> <tbody> <tr> <td>☒ Meta Tag</td><td>☐ DEVICE</td></tr> <tr> <td>☒ Navigational</td><td>☐ BROWSER -INSTANCE</td></tr> <tr> <td>☐ Windows External</td><td>☒ BROWSER -GLOBAL</td></tr> </tbody> </table>	SYNTAX	SCOPE	☒ Meta Tag	☐ DEVICE	☒ Navigational	☐ BROWSER -INSTANCE	☐ Windows External	☒ BROWSER -GLOBAL
SYNTAX	SCOPE								
☒ Meta Tag	☐ DEVICE								
☒ Navigational	☐ BROWSER -INSTANCE								
☐ Windows External	☒ BROWSER -GLOBAL								

gswbrowse_POWERON

The GSWBROWSE_POWERON command specifies an action to initiate when the device resumes power after a suspension.

Command: gswbrowse_PowerOn

Parameter(content): JavaScript Function | URL

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_PowerOn" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_PowerOn,content";
document.location = "gswac:gswbrowse_PowerOn,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_PowerOn,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_PowerOn,content">
```

Where

Content A JavaScript function to execute or a page to navigate when the power resumes.

JavaScript Target

When the Meta Tag is processed the JavaScript function is configured to be called as the device powers up and resumes operation after being suspended.

Example:

```
<META HTTP-EQUIV="gswbrowse_PowerOn" CONTENT="Javascript:OnResume();">
```

URL Target



When the Meta Tag is processed the URL is configured to be called as the device powers up and resumes operation after being suspended.

```
< META HTTP-EQUIV="gswbrowse_PowerOn" CONTENT="http://acme_server/resume.htm">
```

Example: GSWBROWSE_POWERON

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_OnSuspend" CONTENT="Javascript:OnSuspend();">
<META HTTP-EQUIV="gswbrowse_PowerOn" CONTENT="Javascript:OnResume();">
<script>
function OnSuspend()
{
    document.getElementById("suspenddiv").innerHTML="Suspended";
}
function OnResume()
{
    document.getElementById("resumeddiv").innerHTML="Resumed";
}
</script>
</head>
<body>
<p>
Suspended? <div id="suspenddiv">not reached yet...</div>
Resumed? <div id="resumeddiv">not reached yet...</div>
</p>
</body>
</html>
```



Scanner Operation

Category: Scanner

GSWBROWSE_SCANNERNAVIGATE

SYNTAX

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

gswbrowse_SCANNERNAVIGATE

The GSWBROWSE_SCANNERNAVIGATE command takes scanner input and allows processing through a JavaScript Function or URL.

Command: gswbrowse_ScannerNavigate

Parameter(content): JavaScript Function | URL

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_ScannerNavigate" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_ScannerNavigate,content";
document.location = "gswac:gswbrowse_ScannerNavigate,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_ScannerNaviagte,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_ScannerNavigate,content">
```

Where

URL Target

If a URL is specified then it is parsed and the pattern replacement described below will occur before the actual navigation takes place. The pattern specifies the name of the Error information of interest.

If '**errno=%s**' is found then %S will be replaced with the error number from the operating system.

If '**msg=%s**' is found then %S will be replaced with the error message from the operating system.



Example: GSWBROWSE_SCANNERNavigate

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<meta http-equiv="gswbrowse_ScannerNavigate" content="javascript:OnScan('%s', '%s', '%s',
's', '%s');">
<script language=javascript>
function OnScan(a,b,c,d,e)
{
    document.getElementById("scandiv").innerHTML=
        a + " " + b + " " + c + " " + d + " " + e;
}
</script>
</head>
<body>
<p>Scan data, see result below...</p>
Scan: <div id="scandiv">junk</div>
</body>
</html>
```



Timer-based Actions

Category: Timer Based Actions					
GSWBROWSE_TIMERNAVIGATE	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☒ Meta Tag ☒ Navigational ☐ Windows External </td><td> ☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL
SYNTAX	SCOPE				
☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL				

gswbrowse_TIMERNAVIGATE

The GSWBROWSE_TIMERNAVIGATE command launches a JavaScript Function or URL at specified intervals. The current time is also passed as string to the JavaScript Function or URL. **The time interval must** be specified with the GSWBROWSE_TIMERINTERVAL command for this command to work.

Command: gswbrowse_TimerNavigate

Parameter (content): JavaScript Function | URL

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_TimerNavigate" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_TimerNavigate,content";
document.location = "gswac:gswbrowse_TimerNavigate,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_TimerNavigate,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_TimerNavigate,content">
```

Where

Content is a JavaScript function
 Example: **Javascript:TimerTest('%s');**
 OR a URL
 Example: **http://acme_server/timerexpired?time=%s**



- Timer Interval Must Be Specified when using Meta Tag or Navigation Syntax
- gswbrowse_TimerNavigate does not execute until the Timer Interval expires
- Reloading the page cancels gswbrowse_TimerNavigate and gswbrowse_TimerInterval
- Navigation away from the page cancels gswbrowse_TimerNavigate and gswbrowse_TimerInterval

Return

The current time is passed to the JavaScript Function or URL.

Time

Current time in the format **hh:mm:ss**

JavaScript Pseudo-URL Target

When the Meta Tag is processed the JavaScript function will be called with the current time passed to the function.

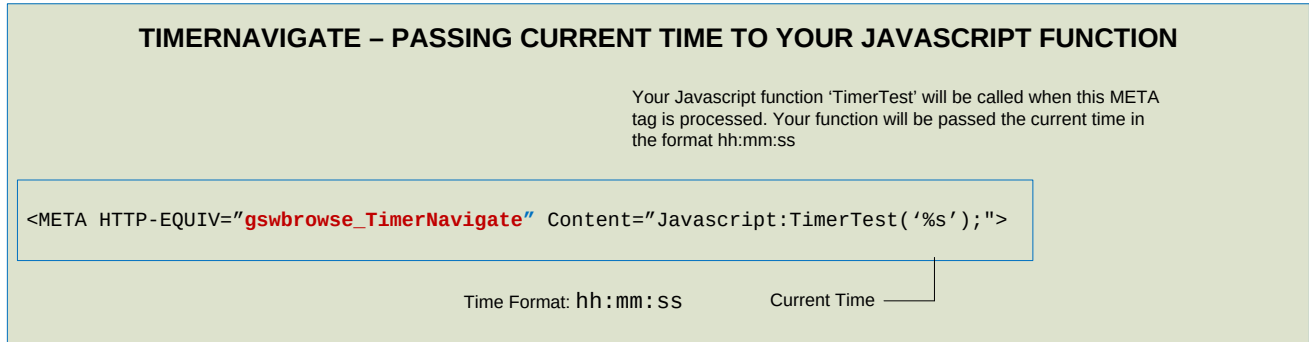


Figure 46: Timer Navigate - Passing the time to your JavaScript function

URL Target

If a URL is specified then it is parsed and the pattern replacement described below will occur before the actual navigation takes place. The pattern specifies the name of the of the signal information of interest.

If 'time=%s' is found then %s will be replaced with the current time value in the format

hh:mm:ss

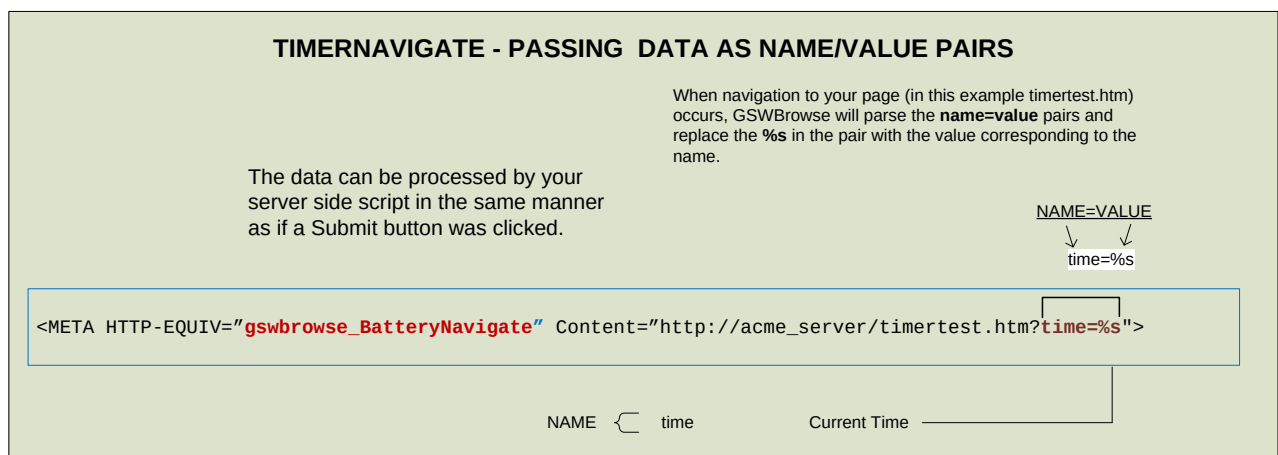


Figure 47: Timer Navigate - Passing the time to you Form (URL).

Example of gswbrowse_timernavigate on [page 81](#)



Category: Timer Based Actions

GSWBROWSE_TIMERINTERVAL**SYNTAX**

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

gswbrowse_TIMERINTERVAL

The GSWBROWSE_TIMERINTERVAL command specifies the time interval (in milliseconds) that the GSWBROWSE_TIMERNAVIGATE command is launched.

Command: gswbrowse_TimerInterval

Parameter(content): Decimal number between 0 and 4294967275
Units: milliseconds

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_TimerInterval" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_TimerInterval,content";
document.location = "gswac:gswbrowse_TimerInterval,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_TimerInterval,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_TimerInterval,content">
```

Where

Content time in milliseconds with a range between 0 and 4294967275.
Commas are not allowed



- Time is in decimal milliseconds (1/1000 of a second)
- Reloading the page cancels gswbrowse_TimerNavigate and gswbrowse_TimerInterval
- Navigation away from the page cancels gswbrowse_TimerNavigate and gswbrowse_TimerInterval



Example: GSWBROWSE_TIMERNAVIGATE

Example: GSWBROWSE_TIMERINTERVAL

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="TimerNavigate" CONTENT="Javascript:TimerTestfun('%s');">
<META HTTP-EQUIV="TimerInterval" CONTENT="1000">
<script id="script_timer_funs" type="text/javascript">
function TimerTestfun(a)
{
    document.getElementById("timerdiv").innerHTML = a;
}
</script>
</head>

<body>
Current time: <div id="timerdiv">junk</div>
</body>
</html>
```



Browser Navigation Errors

Category: Browser Navigation Errors

GSWBROWSE_ERRORNAVIGATE

SYNTAX

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

gswbrowse_ERRORNAVIGATE

The GSWBROWSE_ERRORNAVIGATE command launches a JavaScript Function or URL when a navigation error is detected by GSWBrowse. The error number and/or error message are passed to the target JavaScript or URL.

Command: gswbrowse_ErrorNavigate

Parameter(content): JavaScript Function | URL

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_ErrorNavigate" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_ErrorNavigate,content";
document.location = "gswac:gswbrowse_ErrorNavigate,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_ErrorNavigate,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_ErrorNavigate,content">
```

Where

Content is a JavaScript function

Example: **Javascript:alert('%s', '%s');**

OR a URL

Example:

```
<META HTTP-EQUIV="gswbrowse_ErrorNavigate" CONTENT="
http://acme_server/page_err.htm?errorno=%s&msg=%s">
```

Return

The Operating System Error Number and Error Message are passed to the JavaScript Function or URL.

Error Number

Error Number as defined by the operating system

Error Message

Error Message as defined by the operating system



Some servers display a default/preconfigured page in case a requested page is not found. As a consequence the server does not send error notification to the client(GSWBrowse) and processing of this tag will not occur.

JavaScript Pseudo-URL Target

When the Meta Tag is processed the JavaScript function will be called with the Error Number and Error Message passed to the function.

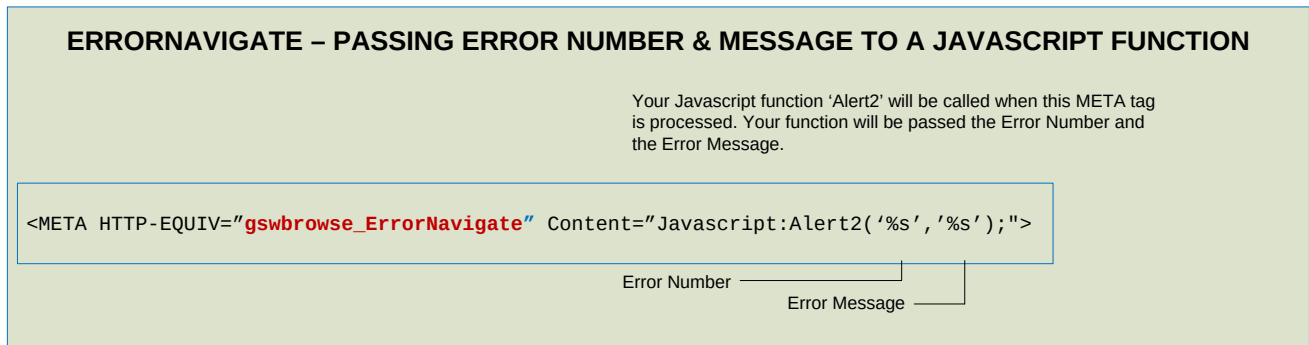


Figure 48: Error Navigate - Passing the OS Error Number and Error Message to your JavaScript function

URL Target

If a URL is specified then it is parsed and the pattern replacement described below will occur before the actual navigation takes place. The pattern specifies the name of the Error information of interest.

If 'errno=%s' is found then %s will be replaced with the error number from the operating system.

If 'msg=%s' is found then %s will be replaced with the error message from the operating system.

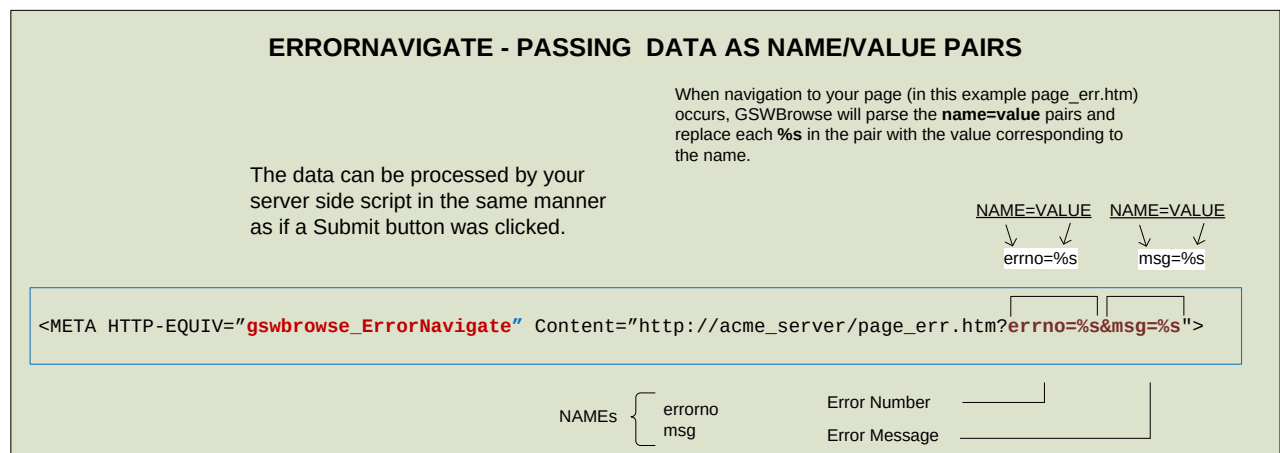


Figure 49: Error Navigate - Passing the Error Number and Message to a URL



Example: GSWBROWSE_ERRORNAVIGATE

```
<html>

<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_ErrorNavigate"
CONTENT="Javascript:alert2('%s','%s');">
<script id ="script_alert2" type="text/javascript">
function alert2(errno,msg)
{
    alert(errno + "-->" + msg);
}
</script>
</head>

<body>
</body>
</html>
```



Exiting Browser

Category: Browser Exiting					
GSWBROWSE_COMMAND	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☒ Meta Tag ☒ Navigational ☐ Windows External </td><td> ☐ DEVICE ☐ BROWSER -INSTANCE ☒ BROWSER -GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☐ BROWSER -INSTANCE ☒ BROWSER -GLOBAL
SYNTAX	SCOPE				
☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☐ BROWSER -INSTANCE ☒ BROWSER -GLOBAL				

gswbrowse_COMMAND

The GSWBROWSE_COMMAND command exits GSWBrowse.

Command: gswbrowse_COMMAND

Parameter(content): Exit

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_Command" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_Command,content";
document.location = "gswac:gswbrowse_Command,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_Command,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_Command,content">
```

Where

Content EXIT

Example:

```
<META HTTP-EQUIV="gswbrowse_Command" CONTENT="Exit">
```



- The GSWBrowse_Application and GSWBrowse_Command commands perform the same function. They are both included for compatibility.

Example: GSWBROWSE_COMMAND

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_command" CONTENT="exit">
</head>
<body>
You should not see this text.
</body>
</html>
```



Category: Browser Exiting

GSWBROWSE_APPLICATION**SYNTAX**

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☐ DEVICE
- ☐ BROWSER -INSTANCE
- ☒ BROWSER -GLOBAL

gswbrowse_APPLICATION

The GSWBROWSE_APPLICATION command exits GSWBrowse.

Command: gswbrowse_APPLICATION

Parameter(content): Quit

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_Application" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_Application,content";
document.location = "gswac:gswbrowse_Command,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_Application,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_Application,content">
```

Where

Content Quit

Example:

```
<META HTTP-EQUIV="gswbrowse_Application" CONTENT="Quit">
```



- The GSWBrowse_Application and GSWBrowse_Command commands perform the same function. They are both included for compatibility.

Example: GSWBROWSE_APPLICATION

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_Application" CONTENT="Quit">
</head>
<body>
You should not see this text.
</body>
```



Soft Input Panel (SIP)

Category: Input Panel	
GSWBROWSE_SIP	SYNTAX ☒ Meta Tag ☒ Navigational ☐ Windows External
	SCOPE ☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL

gswbrowse_SIP

The GSWBROWSE_SIP command controls the display of the Soft Input Panel(SIP). The SIP can be hidden or displayed.

Command: gswbrowse_SIP

Parameter(content): SHOW|HIDE

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_SIP" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_SIP,content";
document.location = "gswac:gswbrowse_SIP,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_SIP,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_SIP,content">
```

Where

Content SHOW|HIDE

SHOW	- Displays the Soft Input Panel
HIDE	- Hides the Soft Input Panel

Example:

```
<META HTTP-EQUIV="gswbrowse_SIP" CONTENT="show">
```

Example: location.href = "gswac:gswbrowse_sip,hide";



Example: GSWBROWSE_SIP

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_SIP" CONTENT="Hide">
</head>
<body>
You should not see SIP when this page loads!!!
<p>
<a href="gswac:gswbrowse_sip,Show">SIP ON</a>
</p>
<p>
<a href="gswac:gswbrowse_sip,Hide">SIP OFF</a>
</p>
</body>
```



Category: Input Panel					
GSWBROWSE_SIPUP	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☒ Meta Tag ☒ Navigational ☐ Windows External </td><td> ☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL
SYNTAX	SCOPE				
☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL				

gswbrowse_SIPUP

The GSWBROWSE_SIPUP command displays the Soft Input Panel(SIP).

Command: gswbrowse_SIPUP

Parameter(content): Empty String

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_SIPUp" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_SIPUp, content";  
document.location = "gswac:gswbrowse_SIPUp, content";
```

Methods

```
windows.navigate("gswac:gswbrowse_SIPUp, content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_SIPUp, content">
```

Where

Content There is no content needed with the SIPUp but the empty string must be provided.

Example:

```
<META HTTP-EQUIV="gswbrowse_SIPUp" CONTENT="">
```

Example: location.href = "gswac:gswbrowse_SIPUp, ";



Example: GSWBROWSE_SIPUP

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_SIP" CONTENT="Hide">
</head>
<body>
You should not see SIP when this page loads!!!
<p>
<a href="gswac:gswbrowse_SipUp,">SIP ON</a>
</p>
<p>
<a href="gswac:gswbrowse_Sip,Hide">SIP OFF</a>
</p>
</body>
```



Category: Input Panel					
GSWBROWSE_MOVESIP	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☒ Meta Tag ☒ Navigational ☐ Windows External </td><td> ☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL
SYNTAX	SCOPE				
☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL				

gswbrowse_MOVESIP

The GSWBROWSE_MOVESIP command moves the Soft Input Panel (SIP) to a specified location using x and y coordinates.

Command: gswbrowse_MOVESIP

Parameter(content): [x coordinate | y coordinate] |
[x coordinate y coordinate]

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_MOVESIP" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_MOVESIP,content";
document.location = "gswac:gswbrowse_MOVESIP,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_MOVESIP,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_MOVESIP,content">
```

Where

Content

A number specifying the x and y screen coordinates for the location of the top left corner of the Soft Input Panel (SIP). The x coordinate (the column) is specified with the syntax "x=n" where n is the x coordinate value. The y coordinate (the row) is specified with the syntax "y=n" where n is the y coordinate value. The top-left corner of the display is defined as coordinate (0,0). The range of the coordinates is dependent on the usable display size. The coordinates are specified in pixels.

The coordinates include the menu bar, address bar, etc. areas of GSWBrowse. One or both coordinates can be specified per command. Usually both will be specified but specification of individual coordinates are provided for compatibility.



- You can move the Soft Input Panel while it is hidden
It will stay hidden until a gswbrowse_SIPUp or a gswbrowse_SIP, SHOW is performed.
- Be careful not to use coordinates that will move the SIP out of the usable display area.

Example:

```
<META HTTP-EQUIV="gswbrowse_MOVESIP" CONTENT="x=0 y=0">
```

Example:

```
location.href = "gswac:gswbrowse_MOVESIP,x=100 y=100";
```

Example:

```
<META HTTP-EQUIV="gswbrowse_MOVESIP" CONTENT="x=0">
```

```
<META HTTP-EQUIV="gswbrowse_MOVESIP" CONTENT="y=0">
```

Example:

```
location.href = "gswac:gswbrowse_MOVESIP,x=100";
```

```
location.href = "gswac:gswbrowse_MOVESIP,y=100";
```



- No spaces before or after the equals sign. **Incorrect** - Content="x = 0 y = 0" **Correct** - Content="x=0, y=0"
- Only space allowed between coordinates. **Incorrect** - Content="x=0, y = 0" **Correct** - Content="x=0 y=0"
- Order of coordinates does not matter. **Correct** - Content="x=0 y=0" **Correct** - Content="y=0 x=0"

No Spaces
Allowed Here

No Comma
Allowed Here

Figure 50: SIPUp Coordinate Flexible Syntax Specifications

Example: GSWBROWSE_SIPUP

```
<head>
```

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

```
<META HTTP-EQUIV="gswbrowse_SipUp" CONTENT="">
```

```
<META HTTP-EQUIV="gswbrowse_MoveSIP" CONTENT="x=0">
```

```
<META HTTP-EQUIV="gswbrowse_MoveSIP" CONTENT="y=0">
```

```
</head>
```

```
<body>
```

```
The SIP should have moved to (0,0)
```

```
</body>
```



Time and Date

Category: Setting Time and Date									
GSWBROWSE_SETTIME	<table border="1"> <thead> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> </thead> <tbody> <tr> <td>☒ Meta Tag</td><td>☐ DEVICE</td></tr> <tr> <td>☒ Navigational</td><td>☐ BROWSER -INSTANCE</td></tr> <tr> <td>☐ Windows External</td><td>☒ BROWSER -GLOBAL</td></tr> </tbody> </table>	SYNTAX	SCOPE	☒ Meta Tag	☐ DEVICE	☒ Navigational	☐ BROWSER -INSTANCE	☐ Windows External	☒ BROWSER -GLOBAL
SYNTAX	SCOPE								
☒ Meta Tag	☐ DEVICE								
☒ Navigational	☐ BROWSER -INSTANCE								
☐ Windows External	☒ BROWSER -GLOBAL								

gswbrowse_SETTIME

The GSWBROWSE_SETTIME command is used to set the System Time on the device.

Command: gswbrowse_SETTIME

Parameter(Content): hh:mm|hh.mm

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_SETTIME" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_SETTIME,content";
document.location = "gswac:gswbrowse_SETTIME,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_SETTIME,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_SETTIME,content">
```

Where

Content The time where **hh** is the hour and the **mm** is the minutes. Use leading zero if hour and/or minutes are less than 10.

Example: Meta Tag Syntax

```
<META HTTP-EQUIV="gswbrowse_SETTIME" CONTENT="19:20">
```

Example: Navigation Syntax

```
document.location = "gswac:gswbrowse_SETTIME,"09:20";
document.location = "gswac:gswbrowse_SETTIME,"19:05";
```

See example on [page 95](#)



Category: Setting Time and Date

GSWBROWSE_SETDATE**SYNTAX**

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☐ DEVICE
- ☐ BROWSER -INSTANCE
- ☒ BROWSER -GLOBAL

gswbrowse_SETDATE

The GSWBROWSE_SETDATE command is used to set the System Date on the device.

Command: gswbrowse_SETDATE

Parameter (Content): mm/dd/yyyy | mm, dd, yyyy | mm-dd-yyyy

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_SETDATE" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_SETDATE,content";
document.location = "gswac:gswbrowse_SETDATE,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_SETDATE,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_SETDATE,content">
```

Where

Content The date where **mm** is the month and the **dd** is the day and **yyyy** is the year. Use leading zero for month and/or day if less than 10.

Example: Meta Tag Syntax

```
<META HTTP-EQUIV="gswbrowse_SETDATE" CONTENT="09/13/2011">
```

Example: Navigation Syntax

```
document.location = "gswac:gswbrowse_SETDATE,09-13-2011";
```



Example: GSWBROWSE_SETDATE

Example: GSWBROWSE_SETTIME

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_SetDate" CONTENT="09/13/2011">
<META HTTP-EQUIV="gswbrowse_SetTime" CONTENT="19:20">
</head>
<body>
You should see date and time set to 09/13/2011 19:20 (GMT)
</body>
```



Font Size Control

Category: Font Size Control	
GSWBROWSE_FONTSIZE	SYNTAX ☒ Meta Tag ☒ Navigational ☐ Windows External
	SCOPE ☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL

gswbrowse_FONTSIZE

The GSWBROWSE_FONTSIZE changes the Font Size relative to the size specified in the HTML content.

Command: gswbrowse_HomeKey

Parameter(Content): Smallest|Smaller|Medium|Larger|Largest

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_FontSize" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_FontSize,content";
document.location = "gswac:gswbrowse_FontSize,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_FontSize,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_FontSize,content">
```

Where

Content Specifies the size of the font relative to the size specified in the HTML content..

Example: Meta Tag Syntax

```
<META HTTP-EQUIV="gswbrowse_FontSize" CONTENT="Smallest">
```

Example: Navigation Syntax

```
document.location = "gswac:gswbrowse_FontSize,Smallest";
```



Example: GSWBROWSE_FONTSIZE

Example 1. Modify to the largest font size

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_FontSize" CONTENT="largest">
</head>
<body>
You should see this in the largest font.
<p>
<a href="font_min.htm">Use smallest font</a>
</p>
</body>
```

Example 2. Modify to the smallest font size

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_FontSize" CONTENT="smallest">
</head>
<body>
You should see this in the smallest font.
<p>
<a href="font_max.htm">Use largest font</a>
</p>
</body>>
```



Cursor Positioning

Category: Cursor Position

GSWBROWSE_CURSORPOS

SYNTAX

- ☒ Meta Tag
- ☒ Navigational
- ☐ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

gswbrowse_CURSORPOS

The GSWBROWSE_CURSORPOS command moves the cursor position specified location using x and y coordinates.

Command: gswbrowse_CursorPos

Parameter(Content): [x coordinate | y coordinate] |
[x coordinate y coordinate]

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_CursorPos" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_CursorPos,content";  
document.location = "gswac:gswbrowse_CursorPos,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_CursorPos,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_CursorPos,content">
```

Where

Content

A number specifying the x and/or y screen coordinate where to place the cursor. The x coordinate (the column) is specified with the syntax "**x=n**" where **n** is the **x** coordinate value. The y coordinate (the row) is specified with the syntax "**y=n**" where **n** is the **y** coordinate value.

The top-left corner of the display is coordinate (0,0). The range of the coordinates is dependent on the usable display size. One or both coordinates can be specified per command. Usually both will be specified but specification of individual coordinates is provided for compatibility. The coordinates are specified in pixels.

Example:

```
<META HTTP-EQUIV="gswbrowse_CursorPos" CONTENT="x=0 y=0">
```

Example: location.href = "gswac:gswbrowse_CursorPos,x=0 y=0";

Example:

```
<META HTTP-EQUIV="gswbrowse_CursorPos" CONTENT="x=0">
```

```
<META HTTP-EQUIV="gswbrowse_CursorPos" CONTENT="y=0">
```

Example:

```
location.href = "gswac:gswbrowse_CursorPos,x=0";
```

```
location.href = "gswac:gswbrowse_CursorPos,y=0";
```

No Spaces
Allowed Here

↓ ↓ ↓ ↓



- No spaces before or after the equals sign. **Incorrect** - Content="x = 0 y = 0" **Correct** - Content="x=0, y=0"
- Only space allowed between coordinates. **Incorrect** - Content="x=0, y = 0" **Correct** - Content="x=0 y=0"

No Comma
Allowed Here

↑

- Order of coordinates does not matter. **Correct** - Content="x=0 y=0" **Correct** - Content="y=0 x=0"

Figure 51: CURSORPOS Coordinate Flexible Syntax Specification

Example: GSWBROWSE_CURSORPOS

```
<<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_CursorPos" CONTENT="x=100">
<META HTTP-EQUIV="gswbrowse_CursorPos" CONTENT="y=100">
</head>
<body>
You should see cursor at (100,100)
</body>>
```



Rebooting Device – Warm/Soft

Category: Reboot Device	
GSWBROWSE_REBOOT	SYNTAX
	☒ Meta Tag ☒ Navigational ☐ Windows External
	SCOPE
	☐ DEVICE ☐ BROWSER -INSTANCE ☒ BROWSER -GLOBAL

gswbrowse_REBOOT

The GSWBROWSE_REBOOT command forces the device to reboot.

Command: gswbrowse_CursorPos

Parameter(Content): cold | warm

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_Reboot" CONTENT="content">
```

NAVIGATION SYNTAX*Properties*

```
Location.href = "gswac:gswbrowse_Reboot,content";  
document.location = "gswac:gswbrowse_Reboot,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_Reboot,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_Reboot,content">
```

Where

Content Warm boot restarts the operating system. Any unsaved data will be lost.

Cold Boot will restore device to vendor defaults. Special programming is needed for third party applications to survive and be restored after a cold boot. Specific behavior with respect to what is lost and what is not is vendor dependent. You may lose registry settings and your data. **Be sure you know what you are doing before initiating a cold boot. Consult vendor documentation.**

Both Cold and Warm Reboots will not preserve contents of the volatile memory. For example - Context of running application when the reboot was initiated, this includes any user data.

Example:

```
<META HTTP-EQUIV="gswbrowse_Reboot" CONTENT="cold">
```

Example:

```
location.href = "gswac:gswbrowse_Reboot,cold";
```



Example: GSWBROWSE_REBOOT

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_Reboot" CONTENT="warm">
</head>
<body>
You should see warm reboot. You will not see this page
<p>
</p>
</body>
```



Send Menu Commands

Category: APP					
SENDMENUCOMMAND	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☐ Meta Tag ☐ Navigational ☒ Windows External </td><td> ☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL				

window.external.app.sendmenucommand

SendMenu is a method of the **window.external.app** object. Programmatically execute any GSWBrowse menu option. The command is interpreted in the context of the active session, exactly as if the user entered the command from the device.

Method: SendMenuCommand

Data Type: NUMBER

Return There is no return value

Action: The menu option selected executes.

EXTERNAL SYNTAX

window.external.app.**SendMenuCommand(content)**

IDL DEFINITION

```
[id(9), helpstring("method SendMenuCommand")]
HRESULT SendMenuCommand(long CommandId)
```

Where
Content

is the numeric identifier of the menu command to execute. The identifier can be the explicit literal (number) or the symbolic name constant for that menu command. If you use symbolic names, the file GSWBrowse.js should be included or the relevant contents copied to your program. See [page 105](#) for the symbolic textual name constants.



- GSWSBrowse Enhanced Feature

Example: SENDMENUCOMMAND 1

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<script language="javascript">
function SendCommand(value)
{
    window.external.app.SendMenuCommand(value);
}
</script>
</head>
<body>
<center>
<table border=5 bordercolor=black cellspacing=8>
<form name="Commander">
<th>
The Command Center
</th>
<tr align="center"><td>
    <input type="text" name="input" maxlength=10 size=24>
</td></tr>
<tr align="center"><td>
    <input type="button" value=" Send Command "
onclick="SendCommand(input.value)">
</td></tr>
<tr align="center"><td>
    <input type="button" value=" BR1 " onclick="SendCommand(32846)">
    <input type="button" value=" BR2 " onclick="SendCommand(32847)">
    <input type="button" value=" BR3 " onclick="SendCommand(32848)">
    <input type="button" value=" BR4 " onclick="SendCommand(32849)">
</td></tr>
</form>
</table>
</body>
```



Example: SENDMENUCOMMAND 2

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<script src="../../scripts/GSWBrowse.js"></script>
<script language="javascript">
function SendCommand(value)
{
    window.external.app.SendMenuCommand(value);
}
</script>
</head>
<body>
<center>
<table border=5 bordercolor=black cellspacing=8>
<form name="Commander">
<th>
The Command Center
</th>
<tr align="center"><td>
    <input type="text" name="input" maxlength=10 size=24>
</td></tr>
<tr align=center><td>
    <input type="button" value=" Send Command "
onclick="SendCommand(input.value)">
</td></tr>
<tr align="center"><td>
    <input type="button" value=" BR1 "
onclick="SendCommand(GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_1)">
    <input type="button" value=" BR2 "
onclick="SendCommand(GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_2)">
    <input type="button" value=" BR3 "
onclick="SendCommand(GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_3)">
    <input type="button" value=" BR4 "
onclick="SendCommand(GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_4)">
    <input type="button" value=" BR5 "
onclick="SendCommand(GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_5)">
</td></tr>
</form>
</table>
</body>
```



Send Menu Symbolic Names and Literals

FILE MENU	Explicit Number	
ID_FILE_OPEN	32783	GSW.GSWBrowse.Menus.ID_FILE_OPEN
ID_FILE_SAVEAS	32796	GSW.GSWBrowse.Menus.ID_FILE_SAVEAS
ID_FILE_PRINT	32770	GSW.GSWBrowse.Menus.ID_FILE_PRINT
ID_FILE_SECURITY	32853	GSW.GSWBrowse.Menus.ID_FILE_SECURITY
ID_FILE_CLOSE	32769	GSW.GSWBrowse.Menus.ID_FILE_CLOSE

Table 2: Send Menu Command – File

EDIT MENU		
ID_EDIT_SELECT_ALL	32794	GSW.GSWBrowse.Menus.ID_EDIT_SELECT_ALL
ID_EDIT_FIND	32771	GSW.GSWBrowse.Menus.ID_EDIT_FIND

Table 3: Send Menu Command – Edit

VIEW MENU		
ID_VIEW_OPTIONS	32773	GSW.GSWBrowse.Menus.ID_VIEW_OPTIONS
ID_VIEW_REGISTRATION	32854	GSW.GSWBrowse.Menus.ID_VIEW_REGISTRATION
ID_VIEW_SET_AS_HOME	32855	GSW.GSWBrowse.Menus.ID_VIEW_SET_AS_HOME
ID_VIEW_RESET_HOME	32859	GSW.GSWBrowse.Menus.ID_VIEW_RESET_HOME
ID_VIEW_ENABLE_BROWSER	32860	GSW.GSWBrowse.Menus.ID_VIEW_ENABLE_BROWSER
ID_VIEW_MULTIPLE_BROWSERS	32858	GSW.GSWBrowse.Menus.ID_VIEW_MULTIPLE_BROWSERS

Table 4: Send Menu Command - View

VIEW MENU - Bars		
ID_VIEW_STATUSBAR	32815	GSW.GSWBrowse.Menus.ID_VIEW_STATUSBAR
ID_VIEW_RESET_BARS	32861	GSW.GSWBrowse.Menus.ID_VIEW_RESET_BARS
ID_VIEW_CAPTION	32827	GSW.GSWBrowse.Menus.ID_VIEW_CAPTION
ID_VIEW_TOOLBAR	32830	GSW.GSWBrowse.Menus.ID_VIEW_TOOLBAR
ID_VIEW_ADDRESSBAR	32828	GSW.GSWBrowse.Menus.ID_VIEW_ADDRESSBAR
ID_VIEW_MENUBAR	32831	GSW.GSWBrowse.Menus.ID_VIEW_MENUBAR
ID_VIEW_TASKBAR	32857	GSW.GSWBrowse.Menus.ID_VIEW_TASKBAR

Table 5: Send Menu Command - View Bars



VIEW MENU - Browsers		
ID_VIEW_BROWSER_1	32846	GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_1
ID_VIEW_BROWSER_2	32847	GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_2
ID_VIEW_BROWSER_3	32848	GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_3
ID_VIEW_BROWSER_4	32849	GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_4
ID_VIEW_BROWSER_5	32850	GSW.GSWBrowse.Menus.ID_VIEW_BROWSER_5

Table 6: Send Menu Command - View Browsers

VIEW MENU - UI Control		Comments
ID_UICONTROL_CTXMENUS	32797	GSW.GSWBrowse.Menus.ID_UICONTROL_CTXMENUS
ID_CONTEXT_MENU_DEFAULT	32836	GSW.GSWBrowse.Menus.ID_CONTEXT_MENU_DEFAULT
ID_CONTEXT_MENU_IMAGE	32837	GSW.GSWBrowse.Menus.ID_CONTEXT_MENU_IMAGE
ID_CONTEXT_MENU_CONTROL	32838	GSW.GSWBrowse.Menus.ID_CONTEXT_MENU_CONTROL
ID_CONTEXT_MENU_TABLE	32839	GSW.GSWBrowse.Menus.ID_CONTEXT_MENU_TABLE
ID_CONTEXT_MENU_TEXTSELECT	32840	GSW.GSWBrowse.Menus.ID_CONTEXT_MENU_TEXTSELECT
ID_CONTEXT_MENU_ANCHOR	32842	GSW.GSWBrowse.Menus.ID_CONTEXT_MENU_ANCHOR
ID_CONTEXT_MENU_UNKNOWN	32843	GSW.GSWBrowse.Menus.ID_CONTEXT_MENU_UNKNOWN
ID_UICONTROL_DIALOG	32806	GSW.GSWBrowse.Menus.ID_UICONTROL_DIALOG
ID_UICONTROL_3DBORDER	32807	GSW.GSWBrowse.Menus.ID_UICONTROL_3DBORDER
ID_UICONTROL_SCROLLBARS	32808	GSW.GSWBrowse.Menus.ID_UICONTROL_SCROLLBARS

Table 7: Send Menu Command - View UI Control

VIEW MENU - Dld Control		Comments
ID_DLCTL_DLIMAGES	32800	GSW.GSWBrowse.Menus.ID_DLCTL_DLIMAGES
ID_DLCTL_VIDEOS	32801	GSW.GSWBrowse.Menus.ID_DLCTL_VIDEOS
ID_DLCTL_BGSOUNDS	32802	GSW.GSWBrowse.Menus.ID_DLCTL_BGSOUNDS
ID_DLCTL_NO_SCRIPTS	32803	GSW.GSWBrowse.Menus.ID_DLCTL_NO_SCRIPTS
ID_DLCTL_NO_JAVA	32804	GSW.GSWBrowse.Menus.ID_DLCTL_NO_JAVA
ID_DLCTL_NO_RUNACTIVEX	32813	GSW.GSWBrowse.Menus.ID_DLCTL_NO_RUNACTIVEX
ID_DLCTL_NO_CLIENTPULL	32833	GSW.GSWBrowse.Menus.ID_DLCTL_NO_CLIENTPULL
ID_DLCTL_NO_FRAMEDOWNLOAD	32834	GSW.GSWBrowse.Menus.ID_DLCTL_NO_FRAMEDOWNLOAD
ID_DLCTL_FORCEOFFLINE	32835	GSW.GSWBrowse.Menus.ID_DLCTL_FORCEOFFLINE

Table 8: Send Menu Command - View Dld Control



GO MENU		
ID_GO_BACK	32774	GSW.GSWBrowse.Menus.ID_GO_BACK
ID_GO_FORWARD	32775	GSW.GSWBrowse.Menus.ID_GO_FORWARD
ID_GO_HOME	32776	GSW.GSWBrowse.Menus.ID_GO_HOM
ID_VIEW_STOP	32780	GSW.GSWBrowse.Menus.ID_VIEW_STO
ID_VIEW_REFRESH	32782	GSW.GSWBrowse.Menus.ID_VIEW_REFRESH

Table 9: Send Menu Command - GO

HELP MENU		
ID_HELP_ABOUT	32778	GSW.GSWBrowse.Menus.ID_HELP_ABOUT

Table 10: Send Menu Command - Help



Printing

Category: Printing					
GSWBROWSE_ZEBRALABEL_PRINT	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☒ Meta Tag ☒ Navigational ☐ Windows External </td><td> ☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL
SYNTAX	SCOPE				
☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL				

gswbrowse_ZEBRALABEL_PRINT

The GSWBROWSE_ZEBRALABEL_PRINT command sends data to the printer. Up to 100,000 bytes of data can be sent to a printer. This includes serial, Bluetooth, IrDA printers and network printers. The completion status can be obtained using the gswbrowse_ZEBRALABEL_COMPLETE tag.

Command: gswbrowse_ZEBRALABEL_PRINT

Parameter(Content): Databytes to print

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_ZebraLabel_Print" CONTENT="content">
```

NAVIGATION SYNTAX

Properties

```
Location.href = "gswac:gswbrowse_ZebraLabel_Print,content";
document.location = "gswac:gswbrowse_ZebraLabel_Print,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_ZebraLabel_Print,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_ZebraLabel_Print,content">
```

Where

Content

Data bytes to send to the printer. Unique to GSWBrowse, up to 100,000 bytes can be sent to the printer.

The text must follow your printer's raw data format, for example CPCL.

The following C/C++ style escape sequences are allowed:

```
\f - formfeed
\t - tab
\n - new line
\r - carriage return
\ooo - octal number
\xHH - hex number (one byte)
\\ - backslash
\0 - zero (one byte)
```



Example:

```
<META HTTP-EQUIV="gswbrowse_ZebraLabel_Print" CONTENT="! 0 200  
200 210 1\r\nTEXT 4 0 30 40 Hello World\r\nFORM\r\nPRINT\r\n">
```

Example:

```
location.href = "gswac:gswbrowse_ZebraLabel_Print, ! 0 200 200  
210 1\r\nTEXT 4 0 30 40 Hello World\r\nFORM\r\nPRINT\r\n";
```

Example: GSWBROWSE_ZEBRALABEL_PRINT

```
<html>  
<head>  
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">  
<meta http-equiv="gswbrowse_ZebraLabel_Complete" content=  
"http://www.yourserver.com/printresult.htm?status=%ld">  
<meta http-equiv="gswbrowse_ZebraLabel_Print"  
content="! 0 200 200 210 1\r\nTEXT 4 0 30 40 Hello  
World\r\nFORM\r\nPRINT\r\n">  
</head>  
<body>  
<p>ZebraLabel_Print test</p>  
<p><a href="default.htm">Return to main print test menu</a></p>  
</body>  
</html>
```



Category: Printing					
GSWBROWSE_ZEBRALABEL_COMPLETE	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☒ Meta Tag ☒ Navigational ☐ Windows External </td><td> ☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL
SYNTAX	SCOPE				
☒ Meta Tag ☒ Navigational ☐ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL				

gswbrowse_ZEBRALABEL_COMPLETE

The GSWBROWSE_ZEBRALABEL_COMPLETE meta tag allows navigation or JavaScript call to occur based on the completion status of the gswbrowse_ZebraLabel_Print meta tag.

Command: gswbrowse_ZEBRALABEL_COMPLETE

Parameter(content): JavaScript Function | URL

META TAG SYNTAX

```
<META HTTP-EQUIV="gswbrowse_ZebraLabel_Complete" CONTENT="content">
```

NAVIGATION SYNTAX

Note: Make sure that the navigational syntax below is executed before printing through gswbrowse_ZebraLabel_Print is initiated.

Properties

```
Location.href = "gswac:gswbrowse_ZebraLabel_Complete,content";
document.location =
"gswwac:gswbrowse_ZebraLabel_Complete,content";
```

Methods

```
windows.navigate("gswac:gswbrowse_ZebraLabelComplete,content");
```

Anchor Tag

```
<a href="gswac:gswbrowse_ZebraLabel_Complete,content">
```

Where

Content

is a JavaScript Pseudo-URL function

Example: content= "**javascript:OnPrint(%ld)**";">

OR

a URL

Example: content= "**http://192.168.1.102/print.htm?status=%ld**">

Where %ld is a placeholder for the printing result. When the call of navigation occurs it will be replaced with '0' or '1'.



Category: Printing					
PRINTSTRING	<table><tr><th>SYNTAX</th><th>SCOPE</th></tr><tr><td>☐ Meta Tag ☐ Navigational ☒ Windows External</td><td>☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL</td></tr></table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL				

`window.external.app.session(idx).printer.PrintString`

PrintString is a method of the `window.external.app.session.printer` object. Send data to the printer

Parameters:

data

Syntax of the data to be sent is described in the [gswwrowse_ZebraLabel_print](#) section.

EXTERNAL SYNTAX

`window.external.app.session(idx).printer.printerstring`

IDL DEFINITION

```
[id(1), helpstring("method PrintString")]  
HRESULT PrintString(BSTR Data);
```

EXAMPLE - Segment

```
var idx = window.external.ThisSessionIndex;  
var Printer = window.external.app.session(idx).Printer;  
Printer.PrintString(document.Commander.info.value);
```



Browser and Session Data

Category: APP									
SESSION	<table border="1"><thead><tr><th>SYNTAX</th><th>SCOPE</th></tr></thead><tbody><tr><td>☐ Meta Tag</td><td>☐ DEVICE</td></tr><tr><td>☐ Navigational</td><td>☒ BROWSER -INSTANCE</td></tr><tr><td>☒ Windows External</td><td>☐ BROWSER -GLOBAL</td></tr></tbody></table>	SYNTAX	SCOPE	☐ Meta Tag	☐ DEVICE	☐ Navigational	☒ BROWSER -INSTANCE	☒ Windows External	☐ BROWSER -GLOBAL
SYNTAX	SCOPE								
☐ Meta Tag	☐ DEVICE								
☐ Navigational	☒ BROWSER -INSTANCE								
☒ Windows External	☐ BROWSER -GLOBAL								

window.external.app.Session

Session is a property of the window.external.app object. It returns the session object corresponding to the provided index of the browser instance.

Property: Session

Input Parameter: Index of the Browser Instance (1-5)

☒ Read ☐ Write

Return The Return value is a session object which contains its properties and methods.

EXTERNAL SYNTAX

window.external.**app.Session(Index)**

IDL DEFINITION

```
[propget, id(5)]  
HRESULT Session(long Index, [out, retval] IDispatch* *pVal); (GSW only)
```



- READ ONLY. Any attempt to set this property will result in a runtime error.

- GSWBrowser Enhanced Feature



Example: WINDOW.EXTERNAL.APP.SESSION

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</head>
<body onload="javascript:fetchvalues();">
<form id="form1" name="form1">
Test of GSWBrowse Object Model<br>
idx=window.external.ThisSessionIndex<input type="text" name="thisidx"
size="40"><br>
window.external.app.session(idx).host<input type="text" name="host" size="40"><br>
window.external.app.session(idx).port<input type="text" name="port" size="40"><br>
window.external.app.session(idx).type<input type="text" name="type" size="40"><br>
window.external.app.session(idx).path<input type="text" name="path" size="40"><br>
window.external.app.session(idx).homepage<input type="text" name="homepage"
size="60"><br>
</form>
function fetchvalues()
{
    var app = window.external.app;
    var idx = window.external.ThisSessionIndex;
    document.form1.thisidx.value = idx;
    var mysession = app.session(idx);
    document.form1.host.value = mysession.host;
    document.form1.port.value = mysession.port;
    document.form1.type.value = mysession.type;
    document.form1.path.value = mysession.path;
    document.form1.homepage.value = mysession.homepage;
}
</script>
</body>
</html>
```



Category: APP

THISSESSIONINDEX**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER INSTANCE
- ☐ BROWSER GLOBAL

window.external.ThisSessionIndex

ThisSessionIndex is a property of the `window.external` object. It provides a number that represents the index of the session containing your execution context. Note: This may or may not be the active browser. This allows a script to find out in which browser instance it is running.

Property: ThisSessionIndex

Data Type: Number

☒ Read ☐ Write

Return The Return value is a NUMBER which represents the Browser Instance Number

Possible Values: 1,2,3,4,5

EXTERNAL SYNTAX

`window.external.ThisSessionIndex`

IDL DEFINITION

```
[propget, id(DISPID_THIS_SESSION_INDEX), helpstring("property  
ThisSessionIndex")]  
HRESULT ThisSessionIndex([out, retval] long *pVal);
```



- READ ONLY. Any attempt to set this property will result in a runtime error.

- GSWBrowser Enhanced Feature



Category: APP

ACTIVSESSIONINDEX**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☐ DEVICE
- ☐ BROWSER INSTANCE
- ☒ BROWSER GLOBAL

`window.external.app.ActiveSessionIndex`

ActiveSessionIndex is a property of the `window.external.app` object. Get/Set the Active Browser.

Property: ActiveSessionIndex

Data Type: Number

☒ Read ☐ Write

Return The Return value is a NUMBER which represents the Active Session.

Possible Values: 1,2,3,4,5

EXTERNAL SYNTAX

`window.external.app.ActiveSessionIndex`

IDL DEFINITION

```
[propget, id(7)]  
HRESULT ActiveSessionIndex([out, retval] long *pVal);
```

```
[propput, id(7)]  
HRESULT ActiveSessionIndex([in] long newVal);
```



- GSWebBrowser Enhanced Feature

Example: WINDOW.EXTERNAL.APP.ACTIVESSESSIONINDEX

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</head>
<body onload="javascript:fetchvalues();">
<form id="form1" name="form1">
Test of GSWebBrowse Object Model<br>
idx=window.external.app.ActiveSessionIndex<input type="text"
name="activesessionidx" size="40"><br>
window.external.app.session(idx).host<input type="text" name="host" size="40"><br>
window.external.app.session(idx).port<input type="text" name="port" size="40"><br>
window.external.app.session(idx).type<input type="text" name="type" size="40"><br>
window.external.app.session(idx).path<input type="text" name="path" size="40"><br>
window.external.app.session(idx).homepage<input type="text" name="homepage"
size="60"><br>
</form>
function fetchvalues()
{
    var app = window.external.app;
    var idx = window.external.app.ActiveSessionIndex;
    document.form1.activesessionidx.value = idx;
    var activesession = app.session(idx);
    document.form1.host.value = activesession.host;
    document.form1.port.value = activesession.port;
    document.form1.type.value = activesession.type;
    document.form1.path.value = activesession.path;
    document.form1.homepage.value = activesession.homepage;

}
</script>
</body>
</html>
```



Category: APP

SESSION(IDX).HOST**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.app.session(idx).host`

Host is a property of the session object. Get the host part of the URL for the specified session.

Property: session.host

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains the Host part of the URL for the session.

EXTERNAL SYNTAX

```
window.external.app.Session(idx).host  
window.external.app.ActiveSession.host
```

IDL DEFINITION

```
[propget, id(1), helpstring("property Host")]  
HRESULT Host([out, retval] BSTR *pVal);
```

Where ***idx*** is the index of the GSWBrowse session.

Example URLs`http://10.0.0.131/gsbrowse/meta/external.htm`

Host Part is: **10.0.0.131**

`http://www.acme.com/gsbrowse/meta/external.htm`

Host Part is: **www.acme.com**

Figure 52: Host part of URL.



- READ ONLY. Any attempt to set this property will result in a runtime error.

- GSWBrowser Enhanced Feature



Example: WINDOW.EXTERNAL.APP.SESSION(IDX).HOST

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</head>
<body onload="javascript:fetchvalues();">
<form id="form1" name="form1">
Test of GSWSBrowse Object Model<br>
idx=window.external.app.ActiveSessionIndex<input type="text"
name="activesessionidx" size="40"><br>
window.external.app.session(idx).host<input type="text" name="host" size="40"><br>
window.external.app.session(idx).port<input type="text" name="port" size="40"><br>
window.external.app.session(idx).type<input type="text" name="type" size="40"><br>
window.external.app.session(idx).path<input type="text" name="path" size="40"><br>
window.external.app.session(idx).homepage<input type="text" name="homepage"
size="60"><br>
</form>
function fetchvalues()
{
    var app = window.external.app;
    var idx = window.external.app.ActiveSessionIndex;
    document.form1.activesessionidx.value = idx;
    var activesession = app.session(idx);
    document.form1.host.value = activesession.host;
    document.form1.port.value = activesession.port;
    document.form1.type.value = activesession.type;
    document.form1.path.value = activesession.path;
    document.form1.homepage.value = activesession.homepage;

}
</script>
</body>
</html>
```



Category: APP					
SESSION(IDX).PORT	<table><tr><th>SYNTAX</th><th>SCOPE</th></tr><tr><td>☐ Meta Tag ☐ Navigational ☒ Windows External</td><td>☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL</td></tr></table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☒ BROWSER -INSTANCE ☐ BROWSER -GLOBAL				

`window.external.app.session(idx).port`

Port is a property of the session object. Get the port number for the specified session.

Property: Session.port

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains the port number for the session.

EXTERNAL SYNTAX

`window.external.app.Session(idx).port`
`window.external.app.Active.port`

IDL DEFINITION

```
[propget, id(1), helpstring("property Host")]  
HRESULT Host([out, retval] BSTR *pVal);
```

Where
idx is the index of the GSWBrowse session.



- READ ONLY. Any attempt to set this property will result in a runtime error.



- GSWebBrowser Enhanced Feature

Example: WINDOW.EXTERNAL.APP.SESSION(INDEX).PORT

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</head>
<body onload="javascript:fetchvalues();">
<form id="form1" name="form1">
Test of GSWebBrowse Object Model<br>
idx=window.external.app.ActiveSessionIndex<input type="text"
name="activesessionidx" size="40"><br>
window.external.app.session(idx).host<input type="text" name="host" size="40"><br>
window.external.app.session(idx).port<input type="text" name="port" size="40"><br>
window.external.app.session(idx).type<input type="text" name="type" size="40"><br>
window.external.app.session(idx).path<input type="text" name="path" size="40"><br>
window.external.app.session(idx).homepage<input type="text" name="homepage"
size="60"><br>
</form>
function fetchvalues()
{
    var app = window.external.app;
    var idx = window.external.app.ActiveSessionIndex;
    document.form1.activesessionidx.value = idx;
    var activesession = app.session(idx);
    document.form1.host.value = activesession.host;
    document.form1.port.value = activesession.port;
    document.form1.type.value = activesession.type;
    document.form1.path.value = activesession.path;
    document.form1.homepage.value = activesession.homepage;
}
</script>
</body>
</html>
```



Category: APP

SESSION(IDX).TYPE**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.app.session(idx).type`

Type is a property of the session object. Get the scheme type of the specified session. Based on the executing content the type will likely be *http* or *https*.

Property: session.type

Data Type: String

☒ Read ☐ Write

Return The return value is a string which contains the scheme type for the session.

EXTERNAL SYNTAX

```
window.external.app.Session(idx).type  
window.external.app.ActiveSession.type
```

IDL DEFINITION

```
[propget, id(1), helpstring("property Host")]  
HRESULT Host([out, retval] BSTR *pVal);
```

Where

Idx is the index of the GSWBrowse session.

Example URLs

`http://10.0.0.131/gsbrowse/meta/external.htm`



Type Part is: **http**

Figure 53: Type part of URL



- READ ONLY. Any attempt to set this property will result in a runtime error.



- GSWebBrowser Enhanced Feature

Example: WINDOW.EXTERNAL.APP.SESSION(IDX).TYPE

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</head>
<body onload="javascript:fetchvalues();">
<form id="form1" name="form1">
Test of GSWebBrowse Object Model<br>
idx=window.external.app.ActiveSessionIndex<input type="text"
name="activesessionidx" size="40"><br>
window.external.app.session(idx).host<input type="text" name="host" size="40"><br>
window.external.app.session(idx).port<input type="text" name="port" size="40"><br>
window.external.app.session(idx).type<input type="text" name="type" size="40"><br>
window.external.app.session(idx).path<input type="text" name="path" size="40"><br>
window.external.app.session(idx).homepage<input type="text" name="homepage"
size="60"><br>
</form>
function fetchvalues()
{
    var app = window.external.app;
    var idx = window.external.app.ActiveSessionIndex;
    document.form1.activesessionidx.value = idx;
    var activesession = app.session(idx);
    document.form1.host.value = activesession.host;
    document.form1.port.value = activesession.port;
    document.form1.type.value = activesession.type;
    document.form1.path.value = activesession.path;
    document.form1.homepage.value = activesession.homepage;
}
</script>
</body>
</html>
```



Category: APP

SESSION(IDX).PATH**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.app.session(idx).path`

Path is a property of the session object. Get the path part of the URL of the specified session.

Property: Session.path

Data Type: String

☒ Read ☐ Write

Return The return value is a string which contains the path for the session.

EXTERNAL SYNTAX

```
window.external.app.Session(idx).path  
window.external.app.ActiveSession.path
```

IDL DEFINITION

```
[[propget, id(4), helpstring("property Path")]]  
HRESULT Path([out, retval] BSTR *pVal);
```

Where

idx is the index of the GSWBrowse session.

Example URL

`http://10.0.0.131/gsbrowse/meta/external.htm`

Path part is: ***/gsbrowse/meta/external.htm***

Figure 54: Path part of URL



- READ ONLY. Any attempt to set this property will result in a runtime error.

- GSWBrowser Enhanced Feature



Example: WINDOW.EXTERNAL.APP.SESSION(IDX).PATH

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</head>
<body onload="javascript:fetchvalues();">
<form id="form1" name="form1">
Test of GSWBrowse Object Model<br>
idx=window.external.app.ActiveSessionIndex<input type="text"
name="activesessionidx" size="40"><br>
window.external.app.session(idx).host<input type="text" name="host" size="40"><br>
window.external.app.session(idx).port<input type="text" name="port" size="40"><br>
window.external.app.session(idx).type<input type="text" name="type" size="40"><br>
window.external.app.session(idx).path<input type="text" name="path" size="40"><br>
window.external.app.session(idx).homepage<input type="text" name="homepage"
size="60"><br>
</form>
function fetchvalues()
{
    var app = window.external.app;
    var idx = window.external.app.ActiveSessionIndex;
    document.form1.activesessionidx.value = idx;
    var activesession = app.session(idx);
    document.form1.host.value = activesession.host;
    document.form1.port.value = activesession.port;
    document.form1.type.value = activesession.type;
    document.form1.path.value = activesession.path;
    document.form1.homepage.value = activesession.homepage;
}
</script>
</body>
</html>
```



Category: APP

SESSION(IDX).HOMEPAGE**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☐ DEVICE
- ☒ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.app.session(idx).homepage`

Homepage is a property of the session object. Get/Set the home page for the specified GSWBrowse instance(tab). Each GSWBrowse session may have different homepages.

Property: session.homepage

Input Parameter: When configuring the homepage this is a String that specified the entire homepage page.

Example:
`http://10.0.0.131/gsbrowse/meta/external.htm`

Data Type: String

☒ Read ☒ Write

Return The return value is a string which contains the home page for this GSWBrowse session.

EXTERNAL SYNTAX

```
window.external.app.Session(idx).homepage  
window.external.app.ActiveSession.homepage
```

IDL DEFINITION

```
[propget, id(5), helpstring("property HomePage")]  
HRESULT HomePage([out, retval] BSTR *pVal);  
  
[propput, id(5), helpstring("property HomePage")]  
HRESULT HomePage([in] BSTR newVal);
```

Where
Idx is the index of the GSWBrowse session.

- GSWBrowser Enhanced Feature

**Example: WINDOW.EXTERNAL.APP.SESSION(IDX).HOMEPAGE**

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</head>
<body onload="javascript:fetchvalues();">
<form id="form1" name="form1">
Test of GSWBrowse Object Model<br>
idx=window.external.app.ActiveSessionIndex<input type="text"
name="activesessionidx" size="40"><br>
window.external.app.session(idx).host<input type="text" name="host" size="40"><br>
window.external.app.session(idx).port<input type="text" name="port" size="40"><br>
window.external.app.session(idx).type<input type="text" name="type" size="40"><br>
window.external.app.session(idx).path<input type="text" name="path" size="40"><br>
window.external.app.session(idx).homepage<input type="text" name="homepage"
size="60"><br>
</form>
function fetchvalues()
{
    var app = window.external.app;
    var idx = window.external.app.ActiveSessionIndex;
    document.form1.activesessionidx.value = idx;
    var activesession = app.session(idx);
    document.form1.host.value = activesession.host;
    document.form1.port.value = activesession.port;
    document.form1.type.value = activesession.type;
    document.form1.path.value = activesession.path;
    document.form1.homepage.value = activesession.homepage;
}
</script>
</body>
</html>
```



Category: GSWBrowse Information					
VERSION	<table><tr><th>SYNTAX</th><th>SCOPE</th></tr><tr><td>☐ Meta Tag ☐ Navigational ☒ Windows External</td><td>☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL</td></tr></table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL				

window.external.app.version

Version is a property of the `window.external.app` object. It provides access to the GSWBrowse release version number.

Property: Version

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains the Release Version number of GSWBrowse.

The Version Number is of the format
x.yy.zzzz

Where

X is the major release number
YY is the minor release number
ZZZZ is the update number

EXTERNAL SYNTAX

`window.external.app.version`

IDL DEFINITION

```
[propget, id(1), helpstring("property Version")]  
HRESULT Version([out, retval] BSTR *pVal);
```

EXAMPLE – segment from [External.htm \(page 226\)](#)

```
var app = window.external.app;  
document.form1.version.value = app.version;
```



• READ ONLY. Any attempt to set this property will result in a runtime error.



Category: APP					
BUILDID	<table><tr><th>SYNTAX</th><th>SCOPE</th></tr><tr><td>☐ Meta Tag ☐ Navigational ☒ Windows External</td><td>☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL</td></tr></table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL				

window.external.app.buildid

BuildId is a property of the window.external.app object. It provides access to the GSWBrowse Build Id which is a unique identifier for each software build.

Property: BuildId

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains the Release Version number of GSWBrowse.

Unique identifier for each version build.

EXTERNAL SYNTAX

window.external.**app.buildid**

IDL DEFINITION

```
[propget, id(2), helpstring("property BuildId")]  
HRESULT BuildId([out, retval] BSTR *pVal);
```

EXAMPLE - segment from [External.htm](#) (page 226)

```
var app = window.external.app;  
document.form1.buildid.value = app.buildid;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Category: APP					
NAME	<table><tr><th>SYNTAX</th><th>SCOPE</th></tr><tr><td>☐ Meta Tag ☐ Navigational ☒ Windows External</td><td>☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL</td></tr></table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL				

window.external.app.name

Name is a property of the window.external.app object. It provides the name of the current application running (GSWBrowse) .

Property: Name

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains the name of the application.

EXTERNAL SYNTAX

window.external.**app.name**

IDL DEFINITION

```
[propget, id(4), helpstring("property UserText")]  
HRESULT UserText(long Index, [out, retval] BSTR *pVal);
```

EXAMPLE - segment from [External.htm](#) ([page 226](#))

```
var app = window.external.app;  
document.form1.name.value = app.name;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Category: APP									
COPYRIGHT	<table><tr><th>SYNTAX</th><th>SCOPE</th></tr><tr><td>☐ Meta Tag</td><td>☐ DEVICE</td></tr><tr><td>☐ Navigational</td><td>☐ BROWSER INSTANCE</td></tr><tr><td>☒ Windows External</td><td>☒ BROWSER GLOBAL</td></tr></table>	SYNTAX	SCOPE	☐ Meta Tag	☐ DEVICE	☐ Navigational	☐ BROWSER INSTANCE	☒ Windows External	☒ BROWSER GLOBAL
SYNTAX	SCOPE								
☐ Meta Tag	☐ DEVICE								
☐ Navigational	☐ BROWSER INSTANCE								
☒ Windows External	☒ BROWSER GLOBAL								

window.external.app.copyright

CopyRight is a property of the window.external.app object. It provides the copyright text for GSWBrowse.

Property: Copyright

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains copyright text for GSWBrowse.

EXTERNAL SYNTAX

window.external.**app.copyright**

IDL DEFINITION

```
[propget, id(6)]  
HRESULT Copyright([out, retval] BSTR *pVal);
```

EXAMPLE – segment from [External.htm](#) (page 226)

```
var app = window.external.app;  
document.form1.name.value = app.copyright;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.

- GSWBrowser Enhanced Feature



Persistent Storage

Category: Persistent Storage					
USERTEXT	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☐ Meta Tag ☐ Navigational ☒ Windows External </td><td> ☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☐ DEVICE ☐ BROWSER INSTANCE ☒ BROWSER GLOBAL				

window.external.app.usertext

UserText is a property of the window.external.app object. UserText is an indexed property that stores user data strings in persistent storage (as defined by the device) shared by all sessions. This can be used for a variety of purposes including passing data between browser sessions.

Property: UserText(index)

Data Type: String

☒ Read ☒ Write

Return Return value is a String which contains the stored user data.

Set Stores the string in persistent storage. The text is stored in GSWebrowse.txt (page 244), so it can be copied to different devices as needed.

Where *Index* is the numeric identifier of the UserText string to read/write. UserText persistent area is limited only by available file storage area.

EXTERNAL SYNTAX

window.external.**app.usertext(index)**

IDL DEFINITION

```
[propget, id(4), helpstring("property UserText")]
HRESULT UserText(long Index, [out, retval] BSTR *pVal);
```

```
[propput, id(4), helpstring("property UserText")]
HRESULT UserText(long Index, [in] BSTR newVal);
```

EXAMPLE – segment from [External.htm](#) (page 226)

```
var app = window.external.app;
app.usertext(15) = "Text 15";
```

```
document.form1.userText15.value = app.userText(15);
```

Category: APP									
ACTIVSESSION	<table border="1"> <thead> <tr> <th style="background-color: #f9f9f9;">SYNTAX</th> <th style="background-color: #f9f9f9;">SCOPE</th> </tr> </thead> <tbody> <tr> <td>☐ Meta Tag</td> <td>☐ DEVICE</td> </tr> <tr> <td>☐ Navigational</td> <td>☐ BROWSER INSTANCE</td> </tr> <tr> <td>☒ Windows External</td> <td>☒ BROWSER GLOBAL</td> </tr> </tbody> </table>	SYNTAX	SCOPE	☐ Meta Tag	☐ DEVICE	☐ Navigational	☐ BROWSER INSTANCE	☒ Windows External	☒ BROWSER GLOBAL
SYNTAX	SCOPE								
☐ Meta Tag	☐ DEVICE								
☐ Navigational	☐ BROWSER INSTANCE								
☒ Windows External	☒ BROWSER GLOBAL								

window.external.app.activesession

ActiveSession is a property of the window.external.app object. It returns the session object associated with the currently visible GSWBrowse instance .

Property: ActiveSession

Data Type: Pointer to the Session Object

☒ Read ☐ Write

Return The return value is the Session Object associated with the active session.

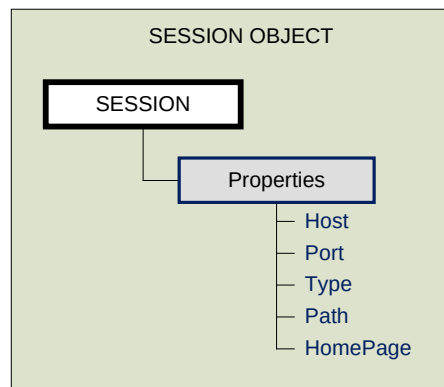


Figure 55: Session Object

EXTERNAL SYNTAX

window.external.**app.activesession**

IDL DEFINITION

```
[propget, id(8), helpstring("property ActiveSession")]
HRESULT ActiveSession([out, retval] IDispatch* *pVal);
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Example: WINDOW.EXTERNAL.APP.ACTIVSESSION

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</head>
<script type="text/javascript">
<body onload="javascript:fetchvalues();">
<form id="form1" name="form1">
Test of GSWBrowse Object Model<br>
window.external.app.activesession.host<input type="text" name="host" size="40"><br>
window.external.app.activesession.port<input type="text" name="port" size="40"><br>
window.external.app.activesession.type<input type="text" name="type" size="40"><br>
window.external.app.activesession.path<input type="text" name="path" size="40"><br>
window.external.app.activesession.homepage<input type="text" name="homepage" size="60"><br>
</form>
function fetchvalues()
{
    var app = window.external.app;
    var activesession = app.activesession;
    document.form1.host.value = activesession.host;
    document.form1.port.value = activesession.port;
    document.form1.type.value = activesession.type;
    document.form1.path.value = activesession.path;
    document.form1.homepage.value = activesession.homepage;

}
</script>
</body>
```



Device Information

Category: DEVICE									
DEVICEID	<table border="1"><thead><tr><th>SYNTAX</th><th>SCOPE</th></tr></thead><tbody><tr><td>☐ Meta Tag</td><td>☒ DEVICE</td></tr><tr><td>☐ Navigational</td><td>☐ BROWSER INSTANCE</td></tr><tr><td>☒ Windows External</td><td>☐ BROWSER GLOBAL</td></tr></tbody></table>	SYNTAX	SCOPE	☐ Meta Tag	☒ DEVICE	☐ Navigational	☐ BROWSER INSTANCE	☒ Windows External	☐ BROWSER GLOBAL
SYNTAX	SCOPE								
☐ Meta Tag	☒ DEVICE								
☐ Navigational	☐ BROWSER INSTANCE								
☒ Windows External	☐ BROWSER GLOBAL								

`window.external.device.deviceid`

DeviceId is a property of the `window.external.device` object. Read the Device Identification String.

Property: Device

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains Device Identification. Device Dependent as specified by Microsoft.

EXTERNAL SYNTAX

`window.external.device.deviceid`

IDL DEFINITION

```
[propget, id(2), helpstring("property DeviceId")]  
HRESULT DeviceId([out, retval] BSTR *pVal);
```

EXAMPLE – segment from [External.htm](#) (page 226)

```
var device = window.external.device;  
document.form1.deviceid.value = device.deviceid;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Category: DEVICE

IPADDRESS**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER INSTANCE
☐ BROWSER GLOBAL

`window.external.device.ipaddress`

IPAddress is a property of the `window.external.device` object. Read the Device IP Address.

Property: Ipaddress

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains Device IP Address. Returned in dotted decimal notation format.

EXTERNAL SYNTAX`window.external.device.ipaddress`**IDL DEFINITION**

```
[propget, id(3), helpstring("property IPAddress")]  
HRESULT IPAddress([out, retval] BSTR *pVal);
```

EXAMPLE – segment from [External.htm](#) (page 226)

```
var device = window.external.device;  
document.form1.ipaddress.value = device.ipaddress;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Category: DEVICE

MACADDRESS**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER INSTANCE
☐ BROWSER GLOBAL

`window.external.device.macaddress`

MACAddress is a property of the `window.external.device` object. Read the Device Media Access Control (MAC) Address.

Property: MACAddress

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains Device MAC Address. Hyphenated format.

EXTERNAL SYNTAX`window.external.device.macaddress`**IDL DEFINITION**

```
[propget, id(4), helpstring("property MACAddress")]
HRESULT MACAddress([out, retval] BSTR *pVal);
```

EXAMPLE - segment from [External.htm](#) (page 226)

```
var device = window.external.device;
document.form1.macaddress.value = device.macaddress;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Category: DEVICE					
PLATFORMID	<table><tr><th>SYNTAX</th><th>SCOPE</th></tr><tr><td>☐ Meta Tag ☐ Navigational ☒ Windows External</td><td>☒ DEVICE ☐ BROWSER INSTANCE ☐ BROWSER GLOBAL</td></tr></table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☒ DEVICE ☐ BROWSER INSTANCE ☐ BROWSER GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☒ DEVICE ☐ BROWSER INSTANCE ☐ BROWSER GLOBAL				

window.external.device.platformid

PLATFORMID is a property of the window.external.device object. Read the Windows CE Platform Identification.

Property: PlatformID

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains Windows CE Platform Id. Device Dependent as specified by Microsoft.

EXTERNAL SYNTAX

window.external.**device.platformid**

IDL DEFINITION

```
[[propget, id(5), helpstring("property PlatformId")]  
HRESULT PlatformId([out, retval] BSTR *pVal);
```

EXAMPLE - segment from [External.htm](#) (page 226)

```
var device = window.external.device;  
document.form1.platformid.value = device.platformid;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Category: DEVICE									
PRESETID	<table border="1"><thead><tr><th>SYNTAX</th><th>SCOPE</th></tr></thead><tbody><tr><td>☐ Meta Tag</td><td>☒ DEVICE</td></tr><tr><td>☐ Navigational</td><td>☐ BROWSER INSTANCE</td></tr><tr><td>☒ Windows External</td><td>☐ BROWSER GLOBAL</td></tr></tbody></table>	SYNTAX	SCOPE	☐ Meta Tag	☒ DEVICE	☐ Navigational	☐ BROWSER INSTANCE	☒ Windows External	☐ BROWSER GLOBAL
SYNTAX	SCOPE								
☐ Meta Tag	☒ DEVICE								
☐ Navigational	☐ BROWSER INSTANCE								
☒ Windows External	☐ BROWSER GLOBAL								

window.external.device.presetid

PRESETID is a property of the window.external.device object. Read the Windows CE Preset Identification.

Property: PresetID

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains Windows CE Preset Id. Device Dependent as specified by Microsoft.

EXTERNAL SYNTAX

window.external.**device.presetid**

IDL DEFINITION

```
[propget, id(6), helpstring("property PresetId")]  
HRESULT PresetId([out, retval] BSTR *pVal);
```

EXAMPLE - segment from [External.htm](#) (page 226)

```
var device = window.external.device;  
document.form1.presetid.value = device.presetid;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



Category: DEVICE									
SERIALNUMBER	<table><tr><th>SYNTAX</th><th>SCOPE</th></tr><tr><td>☐ Meta Tag</td><td>☐ DEVICE</td></tr><tr><td>☐ Navigational</td><td>☐ BROWSER INSTANCE</td></tr><tr><td>☒ Windows External</td><td>☒ BROWSER GLOBAL</td></tr></table>	SYNTAX	SCOPE	☐ Meta Tag	☐ DEVICE	☐ Navigational	☐ BROWSER INSTANCE	☒ Windows External	☒ BROWSER GLOBAL
SYNTAX	SCOPE								
☐ Meta Tag	☐ DEVICE								
☐ Navigational	☐ BROWSER INSTANCE								
☒ Windows External	☒ BROWSER GLOBAL								

window.external.device.serialnumber

SERIALNUMBER is a property of the **window.external.device** object. Read the GSW Product Id which is used for software registration (see [page 4](#)).

Property: SerialNumber

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains Georgia SoftWorks Product ID.

EXTERNAL SYNTAX

window.external.**device.serialnumber**

IDL DEFINITION

```
[propget, id(7), helpstring("property SerialNumber")]  
HRESULT SerialNumber([out, retval] BSTR *pVal);
```

EXAMPLE – segment from [External.htm \(page 226\)](#)

```
var device = window.external.device;  
document.form1.serialnumber.value = device.serialnumber;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.

Category: DEVICE

UUID**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER INSTANCE
☐ BROWSER GLOBAL

`window.external.device.uuid`

UUID is a property of the `window.external.device` object. Read the Universally Unique Identifier (UUID).

Property: UUID

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains UUID. Device Dependent as specified by Microsoft.

EXTERNAL SYNTAX`window.external.device.uuid`**IDL DEFINITION**

```
[propget, id(10), helpstring("property UUID")]  
HRESULT UUID([out, retval] BSTR *pVal);
```

EXAMPLE - segment from [External.htm](#) (page 226)

```
var device = window.external.device;  
document.form1.uuid.value = device.uuid;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.



- GSWBrowser Enhanced Feature



Network Adapter

Category: Network Adapter

NETWORKADAPTERNAME

SYNTAX

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER_INSTANCE
- ☐ BROWSER_GLOBAL

`window.external.device.networkadaptername`

NetworkAdapterName is a property of the `window.external.device` object. Read the network adapter name as returned from the operating system.

Property: UUID

Data Type: String

☒ Read ☐ Write

Return The return value is a String which contains the Network Adapter Name.

EXTERNAL SYNTAX

`window.external.device.networkadaptername`

IDL DEFINITION

```
[propget, id(11)]  
HRESULT NetworkAdapterName([out, retval] BSTR *pVal); (GSW only)
```

EXAMPLE – segment from [External.htm](#) (page 226)

```
var device = window.external.device;  
document.form1.networkadaptername.value = device.networkadaptername;
```



- READ ONLY. Any attempt to set this property will result in a runtime error.

- GSWBrowser Enhanced Feature



Vibrating the Device

Category: Vibrating The Device									
VIBRATE	<table><tr><th>SYNTAX</th><th>SCOPE</th></tr><tr><td>☐ Meta Tag</td><td>☒ DEVICE</td></tr><tr><td>☐ Navigational</td><td>☐ BROWSER INSTANCE</td></tr><tr><td>☒ Windows External</td><td>☐ BROWSER GLOBAL</td></tr></table>	SYNTAX	SCOPE	☐ Meta Tag	☒ DEVICE	☐ Navigational	☐ BROWSER INSTANCE	☒ Windows External	☐ BROWSER GLOBAL
SYNTAX	SCOPE								
☐ Meta Tag	☒ DEVICE								
☐ Navigational	☐ BROWSER INSTANCE								
☒ Windows External	☐ BROWSER GLOBAL								

`window.external.device.vibrate`

Vibrate is a method of the `window.external.device` object. Cause the device to vibrate for a specified duration of time.

Method: Vibrate

Data Type: Number

Where The number value specifies the number of milliseconds to vibrate the device.

EXTERNAL SYNTAX

`window.external.device.vibrate`

IDL DEFINITION

```
[id(14)] HRESULT Vibrate(LONG DurationMS); (GSW only)
```

EXAMPLE - Vibrate

```
function Vibrate()  
{  
    window.external.device.vibrate(1000);  
}
```



- LXE Devices ONLY. This External is functional only on LXE Devices and ignored (no execution error will occur) on other devices as of this writing. Future releases may include other devices.

- GSWBrowser Enhanced Feature



Taskbar Control

Category: Controlling the Taskbar					
SETTASKBARENABLED	<table> <tr> <th>SYNTAX</th><th>SCOPE</th></tr> <tr> <td> ☐ Meta Tag ☐ Navigational ☒ Windows External </td><td> ☒ DEVICE ☐ BROWSER -INSTANCE ☐ BROWSER -GLOBAL </td></tr> </table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☒ DEVICE ☐ BROWSER -INSTANCE ☐ BROWSER -GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☒ DEVICE ☐ BROWSER -INSTANCE ☐ BROWSER -GLOBAL				

window.external.device.settaskbarenable

SetTaskBarEnabled is a method of the window.external.device object. Enable/Disable the taskbar. When disabled the taskbar is not selectable. When disabled, if the taskbar is clicked a beep will be sounded.

Method:	SetTaskBarEnabled
Data Type:	Number
Where	1 Enables, 0 Disables.

EXTERNAL SYNTAX

window.external.**device.settaskbarenable**

IDL DEFINITION

```
[id(15)] HRESULT SetTaskBarEnabled(LONG bEnable);
```

EXAMPLE

```
var TaskBarEnabled = 1;

function ToggleTaskBarEnabled()
{
    var dev = window.external.device;
    if (TaskBarEnabled == 1)
        dev.SetTaskBarEnabled(0);
    else
        dev.SetTaskBarEnabled(1);
    TaskBarEnabled = !TaskBarEnabled;
}
```

- GSW ONLY



Category: Controlling the Taskbar

SETTASKBARVISIBLE**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.settaskbarvisible`

SetTaskBarVisible is a method of the `window.external.device` object. This hides/shows the taskbar. When hidden, `ctrl-esc` will not show the taskbar.

Method:	SetTaskBarVisible
Data Type:	Number
Where	1 Makes Taskbar Visible, 0 Makes Taskbar Invisible

EXTERNAL SYNTAX`window.external.device.settaskbarvisible`**IDL DEFINITION**

```
[id(16)] HRESULT SetTaskBarVisible(LONG bEnable);
```

EXAMPLE

```
//TaskBar will disappear and ctrl-esc will not show it. GSWBrowse will  
//use all of the screen area.
```

```
var TaskBarVisible = 1;
```

```
function ToggleTaskBar()  
{  
    var dev = window.external.device;  
    if (TaskBarVisible == 1)  
        dev.SetTaskBarVisible(0);  
    else  
        dev.SetTaskBarVisible(1);  
    TaskBarVisible = !TaskBarVisible;  
}
```



- GSW ONLY

Category: Controlling the Taskbar

SETSTARTBUTTONVISIBLE**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.setstartbuttonvisible`

SetStartButtonVisible is a method of the `window.external.device` object. This hides/shows the start button.

Method: SetStartButtonVisible

Data Type: Number

Where 1 Makes start button visible,
0 Makes start button invisible

EXTERNAL SYNTAX`window.external.device.SetStartButtonVisible`**IDL DEFINITION**

```
[id(17)] HRESULT SetStartButtonVisible(LONG bEnable);
```

EXAMPLE

```
//the Start button will not disappear ... but the whole TaskBar becomes  
//disabled and visible, no error beep when clicked
```

```
var StartBtnVisible = 1;  
  
function ToggleStartBtn()  
{  
    var dev = window.external.device;  
    if (StartBtnVisible == 1)  
        dev.SetStartButtonVisible(0);  
    else  
        dev.SetStartButtonVisible(1);  
    StartBtnVisible = !StartBtnVisible;  
}
```

- GSW ONLY



- At the time of this writing, in many cases this command does not behave as expected. Perhaps in a future release of Windows this will be corrected.
- GSW observations have been noted in the example.



Category: Controlling the Taskbar

SETSIPBUTTONVISIBLE**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.setsipbuttonvisible`

SetSipButtonVisible is a method of the `window.external.device` object. This hides/shows the Soft Input Panel (SIP) Button.

Method: SetSipButtonVisible

Data Type: Number

Where

- 1 Makes SIP Button Visible
- 0 Makes SIP Button Invisible

EXTERNAL SYNTAX`window.external.device.SetSipButtonVisible`**IDL DEFINITION**

```
[id(18)] HRESULT SetSIPButtonVisible(LONG bEnable);
```

EXAMPLE

```
//this call uses SHFS_SHOWSIPBUTTON and does not do ANYTHING under Windows
// CE 4.2 and 5.0 one day may start working...

var SIPBtnVisible = 1;

function ToggleSIPBtn()
{
    var dev = window.external.device;
    if (SIPBtnVisible == 1)
        dev.SetSIPButtonVisible(0);
    else
        dev.SetSIPButtonVisible(1);
    SIPBtnVisible = !SIPBtnVisible;
}
```



- At the time of this writing, this does not work under Windows CE 4.2 or CE 5.0. Perhaps in a future release of Windows this will be corrected.

- GSWBrowser Enhanced Feature



Vendor API Version

Category: Reading Vendor's API Version Number					
APIVERSION	<table border="1"> <tr> <th>SYNTAX</th> <th>SCOPE</th> </tr> <tr> <td> ☐ Meta Tag ☐ Navigational ☒ Windows External </td> <td> ☒ DEVICE ☐ BROWSER INSTANCE ☐ BROWSER GLOBAL </td> </tr> </table>	SYNTAX	SCOPE	☐ Meta Tag ☐ Navigational ☒ Windows External	☒ DEVICE ☐ BROWSER INSTANCE ☐ BROWSER GLOBAL
SYNTAX	SCOPE				
☐ Meta Tag ☐ Navigational ☒ Windows External	☒ DEVICE ☐ BROWSER INSTANCE ☐ BROWSER GLOBAL				

window.external.device.apiversion

APIVersion is a property of the window.external.device object. It returns the Version Number of the Vendors Application Programming Interface (API).

Property: APIVersion

Data Type: String

☒ Read ☐ Write

Where The Return value is a string that contains the API Version Number.

EXTERNAL SYNTAX

window.external.**device.APIVersion**

IDL DEFINITION

```
[propget, id(19)] HRESULT APIVersion([out, retval] BSTR *pVal);
```

EXAMPLE – segment from [External.htm](#) ([page 226](#))

```
var device = window.external.device;
document.form1.apiversion.value = device.APIVersion;
```



- LXE Devices ONLY. This External is functional only on LXE Devices and returns **Unknown** on other devices as of this writing. Future releases may include other devices.

- GSWBrowser Enhanced Feature

LXE Device API's

Georgia SoftWorks provides specialized interfaces to LXE device API's further extending the programming power of GSWBrowse. With these specialized interfaces the developer gains access to extra device features while maintaining a consistent development style and enjoying the benefits of GSWBrowse.

NOTE: Please see the LXE documentation (provided by LXE) for complete and the most current information.

A technical hurdle is that JavaScript is unable to pass parameters by reference to methods of COM objects⁹. The native LXE API's (provided by LXE in the file lxeapi.dll) use pointers and pass parameters by reference, which makes it impossible to create window.external methods with the same synopsis.

To work around this technical problem GSWBrowse has created a "GSW Helper" COM object which provides functionality analogous to passing parameters by reference with the native LXE API COM object.

GSWBrowse Helper Object provides an area that is used for data exchange with the COM object. The Helper Object is passed to the COM object rather than individual data. After the call, individual data is retrieved from the Helper Object.

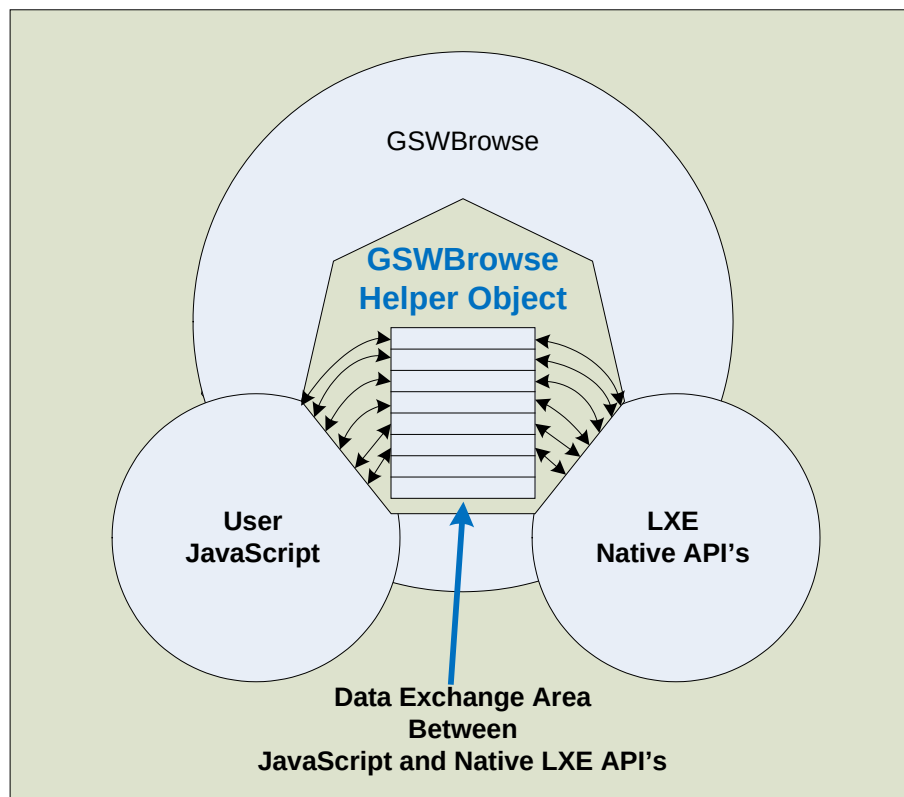


Figure 56: GSWBrowse Helper Object

See [page 216](#) details about the GSW Helper Object and [page 228](#) for an example.

⁹ This is a JavaScript limitation not a COM limitation.

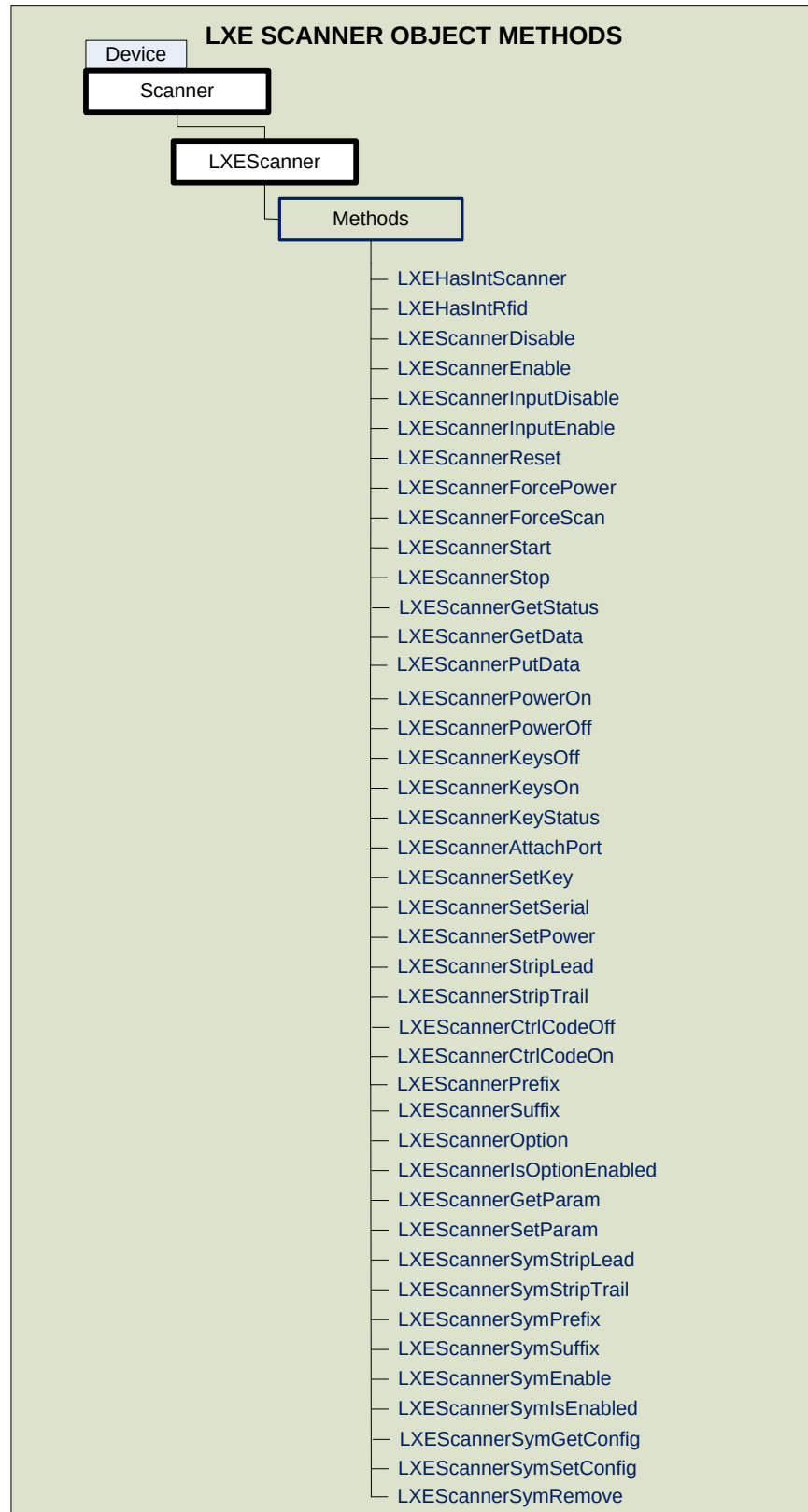


Figure 57: LXE Scanner Object / Methods



Category: LXE Device API's

LXEHASINTSCANNER**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXEHASINTSCANNER

LXEHasIntScanner is a method of the lxescanner object. Determine if the device has an internal scanner.

Parameters**oHelperObject** -GSWBrowse Helper Object (See [page 234](#))**Return Value**

1 Device has Internal Scanner
 0 No Internal Scanner

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEHasIntScanner(oHelperObject)

IDL DEFINITION

```
[id(1), helpstring("method LXEHasIntScanner")]
HRESULT LXEHasIntScanner(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEHasIntScanner(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXEHASINTRFID**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.scanner.lxescanner.LXEHASINTRFID`

LXEHasIntRFID is a method of the `lxescanner` object. Determine if the device has an internal RFID reader.

Parameters**`oHelperObject`** -GSWBrowse Helper Object (See [page 234](#))**Return Value**

1 Device has Internal RFID Reader

0 No Internal RFID Reader

OS ErrorOS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.**EXTERNAL SYNTAX**`window.external.device.scanner.lxescanner.LXEHasIntRFID(oHelperObject)`**IDL DEFINITION**

```
[id(2), helpstring("method LXEHasIntRFID")]  
HRESULT LXEHasIntRFID(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
nResult = lxe.LXEHasIntRFID(oHelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERDISABLE**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.scanner.lxescanner.LXESCANNERDISABLE`

LXEScannerDisable is a method of the lxescanner object. This method causes the scanner driver to disable operation.

Parameters**[oHelperObject](#) -**

GSWBrowse Helper Object (See [page](#) 234)

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

`window.external.device.scanner.lxescanner.LXEScannerDisable\(oHelperObject\)`

IDL DEFINITION

```
[id(3), helpstring("method LXEScannerDisable")]
HRESULT LXEScannerDisable(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerDisable(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERENABLE**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERENABLE

LXEScannerEnable is a method of the lxescanner object. Cause scanner driver to return to normal operation. Wait 1.5 seconds after making this call before attempting to scan.

Parameters

oHelperObject -
GSWBrowse Helper Object (See [page](#) 234)

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerEnable(oHelperObject)

IDL DEFINITION

```
[id(3), helpstring("method LXEScannerEnable")]
HRESULT LXEScannerEnable(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerEnable(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERINPUTDISABLE**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.scanner.lxescanner.LXESCANNERINPUTDISABLE`

LXEScannerInputDisable is a method of the lxescanner object. Used to disable all scan keys and trigger button. This function is used when the delay of 1 or 2 seconds in enabling/disabling the scanner is unacceptable.

Parameters**oHelperObject -**GSWBrowse Helper Object (See [page 234](#))**Return Value**

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX`window.external.device.scanner.lxescanner.LXEScannerInputDisable(oHelperObject)`**IDL DEFINITION**

```
[id(5), helpstring("method LXEScannerInputDisable")]
HRESULT LXEScannerInputDisable(IDispatch *pDisp, [out, retval]LONG
*pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerInputDisable(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERINPUTENABLE**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERINPUTENABLE

LXEScannerInputEnable is a method of the lxescanner object. Used to enable all scan keys and trigger button. This function is used when the delay of 1 or 2 seconds in enabling/disabling the scanner is unacceptable.

Parameters**oHelperObject** -GSWBrowse Helper Object (See [page 234](#))**Return Value**

Returns same numeric value as returned by corresponding native LXE API
 See *LXE documentation for details*.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerInputEnable(oHelperObject)

IDL DEFINITION

```
[id(6), helpstring("method LXEScannerInputEnable")]
HRESULT LXEScannerInputEnable(IDispatch *pDisp, [out, retval]LONG
*pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerInputEnable(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERRESET**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERRESET

LXEScannerReset is a method of the lxescanner object. This method causes the scanner driver to reload all of the registry settings. It also closes and reopens scanner connections as well as reloading all symbology registry configurations.

Parameters**oHelperObject** -GSWBrowse Helper Object (See [page 234](#))**Return Value**

Returns same numeric value as returned by corresponding native LXE API
See *LXE documentation for details*.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerReset(oHelperObject)

IDL DEFINITION

```
[id(5), helpstring("method LXEScannerReset")]
HRESULT LXEScannerReset(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerReset(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERFORCEPOWER**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERFORCEPOWER

LXEScannerForcePower is a method of the lxescanner object. This method forces the serial port to supply power to the scanner.

Parameters**port** - Serial Port

Where valid values for port are:

- | | |
|--|-----|
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_DISABLED | (0) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM1 | (1) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM2 | (2) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM3 | (3) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_INTERNAL | (4) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_RFID | (5) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_BLUETOOTH | (6) |

val - No Power or 5v power on pin

valid values are:

- 0 - no power, pin is Ring Indicator (R1)
 1 - 5v power on pin

oHelperObject -GSWBrowse Helper Object (See [page 234](#))**Return Value**

Returns same numeric value as returned by corresponding native LXE API
 See *LXE documentation for details*.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerForcePower(port, val, oHelperObject)
```

IDL DEFINITION

```
[id(id(8), helpstring("method LXEScannerForcePower"))]
HRESULT LXEScannerForcePower(LONG port, LONG val, IDispatch *pDisp, [out,
retval]LONG *pnResult);
```



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
nResult =  
    lxe.LXEScannerForcePower(GSW.GSWBrowse.Scanner.LXE.SCANPORT_INTERNAL,  
        1, oHelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERFORCESCAN**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.scanner.lxescanner.LXESCANNERFORCESCAN`

LXEScannerForceScan is a method of the lxescanner object. This method forces the serial port to supply power to the scanner.

Parameters**oHelperObject -**

GSWBrowse Helper Object (See [page 234](#))

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

`window.external.device.scanner.lxescanner.LXEScannerForceScan(oHelperObject)`

IDL DEFINITION

```
[id(9), helpstring("method LXEScannerForceScan")]
HRESULT LXEScannerForceScan(IDispatch *pDisp, [out, retval] LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerForceScan(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERSTART**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSTART

LXEScannerStart is a method of the lxescanner object. This method generates a system event that causes the driver to start a scan operation on the internal scanner.

Parameters

oHelperObject -
GSWBrowse Helper Object (See [page](#) 234)

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerStart(oHelperObject)

IDL DEFINITION

```
[id(10), helpstring("method LXEScannerStart")]
HRESULT LXEScannerStart(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerStart(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSTOP**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.scanner.lxescanner.LXESCANNERSTOP`

LXEScannerStop is a method of the `lxescanner` object. This method generates a system event that causes the driver to terminate a scan operation on the internal scanner.

Parameters**`oHelperObject`** -GSWBrowse Helper Object (See [page 234](#))**Return Value**

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX`window.external.device.scanner.lxescanner.LXEScannerStop(oHelperObject)`**IDL DEFINITION**

```
[id(11), helpstring("method LXEScannerStop")]  
HRESULT LXEScannerStop(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
nResult = lxe.LXEScannerStop(oHelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERGETSTATUS**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERGETSTATUS

LXEScannerGetStatus is a method of the lxescanner object. This method generates a system event that causes the driver to terminate a scan operation on the internal scanner.

Parameters

oHelperObject -
GSWBrowse Helper Object (See [page](#) 234)

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.**lxescanner.LXEScannerGetStatus(oHelperObject)**

IDL DEFINITION

```
[id(12), helpstring("method LXEScannerGetStatus")]
HRESULT LXEScannerGetStatus(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerGetStatus(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERGETDATA**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.scanner.lxescanner.LXESCANNERGETDATA`

LXEScannerGetData is a method of the lxescanner object.

Parameters

oHelperObject -

GSWBrowse Helper Object (See [page](#) 234)

Parameter Passing

Int1 ➡ - 'buflen' is passed to the function in Int1 of the GSW Helper Object.

Int1 ⬅ - After the call 'buflen' returned from the API is placed in Int1 member of the GSW Helper Object.

String1 ⬅ - Contents of 'buf' are returned as JavaScript string in String1 member of the GSW Helper Object.

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

`window.external.device.scanner.lxescanner.LXEScannerGetData(oHelperObject)`

IDL DEFINITION

```
[id(13), helpstring("method LXEScannerGetData")]
HRESULT LXEScannerGetData(IDispatch *pDisp, [out, retval]LONG *pnResult);
```



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
oHelperObject.Int1 = 100;
nResult = lxe.LXEScannerGetData(oHelperObject);
var nOSError = oHelperObject.LastError;
var strScannedDataLength = oHelperObject.Int1;
var strScannedData = oHelperObject.String1;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERPUTDATA**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERPUTDATA

LXEScannerPutData is a method of the lxescanner object.**Parameters****strData** - used to pass the 'buf' parameter to the function.**oHelperObject** -
GSWBrowse Helper Object (See [page](#) 234)**Parameter Passing****Int1** ➡ - 'buflen' is passed to the function in Int1 of the GSW Helper Object.**Int1** ⬅ - After the call 'buflen' returned from the API is also in Int1 member of the GSW Helper Object.**Return Value**Returns same numeric value as returned by corresponding native LXE API
*See LXE documentation for details.***OS Error**OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.**EXTERNAL SYNTAX**

window.external.device.scanner.lxescanner.LXEScannerPutData(strData, oHelperObject)

IDL DEFINITION

```
id(14), helpstring("method LXEScannerPutData")
HRESULT LXEScannerPutData(BSTR Data, IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerPutData(strData, oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERPOWERON**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERPOWERON

LXEScannerPowerOn is a method of the lxescanner object. This method generates a system event that causes the driver to apply power to all active scanner devices.

Parameters

oHelperObject -
GSWBrowse Helper Object (See [page](#) 234)

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerPowerOn(oHelperObject)

IDL DEFINITION

```
[id(15), helpstring("method LXEScannerPowerOn")]
HRESULT LXEScannerPowerOn(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerPowerOn(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERPOWEROFF**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERPOWEROFF

LXEScannerPowerOff is a method of the lxescanner object. This method generates a system event that causes the driver to power down all active scanner devices.

Parameters

oHelperObject -

GSWBrowse Helper Object (See [page 234](#))

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerPowerOff(oHelperObject)

IDL DEFINITION

```
id(16), helpstring("method LXEScannerPowerOff")
HRESULT LXEScannerPowerOff(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerPowerOff(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERKEYSOFF**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.scanner.lxescanner.LXESCANNERKEYSOFF`

LXEScannerKeysOff is a method of the `lxescanner` object. This method generates a system event that causes the driver to stop processing scanned data as keystroke messages.

Parameters

oHelperObject -
GSWBrowse Helper Object (See [page 234](#))

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

`window.external.device.scanner.lxescanner.LXEScannerKeysOff(oHelperObject)`

IDL DEFINITION

```
[id(17), helpstring("method LXEScannerKeysOff")]  
HRESULT LXEScannerKeysOff(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
nResult = lxe.LXEScannerKeysOff(oHelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERKEYSON**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERKEYSON

LXEScannerKeysOn is a method of the lxescanner object. This method generates a system event that causes the driver to start processing scanned data as keystroke messages sent to the front most window.

Parameters

oHelperObject -

GSWBrowse Helper Object (See [page](#) 234)

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerKeysOn(oHelperObject)

IDL DEFINITION

```
[id(18), helpstring("method LXEScannerKeysOn")]
HRESULT LXEScannerKeysOn(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerKeysOn(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERKEYSTATUS**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERKEYSTATUS

LXEScannerKeyStatus is a method of the lxescanner object. This method identifies if the scanner wedge is processing data as keystroke messages or being buffered.

Parameters

oHelperObject -
GSWBrowse Helper Object ((See [page](#) 234))

Return Value

1 Processing data as keystroke messages
0 Data is being buffered

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerKeyStatus(oHelperObject)

IDL DEFINITION

```
[id(19), helpstring("method LXEScannerKeyStatus")]
HRESULT LXEScannerKeyStatus(IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerKeyStatus(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERATTACHPORT**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERATTACHPORT

LXEScannerAttachPort is a method of the lxescanner object. This method changes the wedge port attached to the scanner driver as specified and saves it to the registry.

Parameters**port** - Serial Port

Where valid values for port are:

- | | |
|--|-----|
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_DISABLED | (0) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM1 | (1) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM2 | (2) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM3 | (3) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_INTERNAL | (4) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_RFID | (5) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_BLUETOOTH | (6) |

val -

Where valid values for val are:

- | | |
|--------------------------------------|-----|
| GSW.GSWBrowse.Scanner.LXE.WEDGEPORT1 | (1) |
| GSW.GSWBrowse.Scanner.LXE.WEDGEPORT2 | (2) |
| GSW.GSWBrowse.Scanner.LXE.WEDGEPORT3 | (3) |
- (CE 5.x and later)

oHelperObject -GSWBrowse Helper Object (See [page 234](#))**Return Value**

Returns same numeric value as returned by corresponding native LXE API

See *LXE documentation for details*.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerAttachPort(port, val, oHelperObject)
```

IDL DEFINITION

```
[id(20), helpstring("method LXEScannerAttachPort")]
```



```
HRESULT LXEScannerAttachPort(LONG port, LONG val, IDispatch *pDisp, [out,  
retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
var nResult =  
    lxe.LXEScannerAttachPort(GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM1, GSW.GS  
        WBrowse.Scanner.LXE.WEDGEPORT2, oHelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSETKEY**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSETKEY

LXEScannerSetKey is a method of the lxescanner object. This method changes the action of the scan key specified to the action specified.

Parameters**port** -

Where valid values for port are:

- GSW.GSWBrowse.Scanner.LXE.SCANKEYLEFT (1)
- GSW.GSWBrowse.Scanner.LXE.SCANKEYRIGHT (2)
- GSW.GSWBrowse.Scanner.LXE.SCANKEYTRIG (3)
- Handle Trigger

val -

Where valid values for val are:

- GSW.GSWBrowse.Scanner.LXE.SCANKEY_DISABLED (0)
- GSW.GSWBrowse.Scanner.LXE.SCANKEY_SCAN (1)
- Internal Scanner
- GSW.GSWBrowse.Scanner.LXE.SCANKEY_ENTER (2)
- GSW.GSWBrowse.Scanner.LXE.SCANKEY_TAB (3)
- GSW.GSWBrowse.Scanner.LXE.SCANKEY_VIRTKEY (4)
- GSW.GSWBrowse.Scanner.LXE.SCANKEY_FLDEXIT (5)
- GSW.GSWBrowse.Scanner.LXE.SCANKEY_RFID (6)
- GSW.GSWBrowse.Scanner.LXE.SCANKEY_ESC (7)

oHelperObject -GSWBrowse Helper Object (See [page 234](#))**Return Value**

Returns same numeric value as returned by corresponding native LXE API

See *LXE documentation for details*.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerSetKey(port, val,
oHelperObject)
```

IDL DEFINITION

```
[[id(21), helpstring("method LXEScannerSetKey")]]
```



```
HRESULT LXEScannerSetKey(LONG port, LONG val, IDispatch *pDisp, [out,  
retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
var nResult = lxe.LXEScannerSetKey(GSW.GSWBrowse.Scanner.LXE.SCANKEYLEFT,  
GSW.GSWBrowse.Scanner.LXE.SCANKEY_ENTER,oHelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSETSERIAL**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSETSERIAL

LXEScannerSetSerial is a method of the lxescanner object. This method changes the configuration of the scanner serial port on the specified port and saves it to the registry and then reloads the registry.

Parameters**port** -

where valid values for port are:

- | | |
|--|-----|
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_DISABLED | (0) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM1 | (1) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM2 | (2) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM3 | (3) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_INTERNAL | (4) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_RFID | (5) |
| GSW.GSWBrowse.Scanner.LXE.SCANPORT_BLUETOOTH | (6) |

baud - is used to pass the 'baud'

data - is used to pass the 'data'

parity - is used to pass the 'parity'

stop - is used to pass the 'stop bits'

oHelperObject -

GSWBrowse Helper Object (See [page](#) 234)

Return Value

Returns same numeric value as returned by corresponding native LXE API
See *LXE documentation for details*.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerSetSerial(port, baud, data, parity, stop, oHelperObject)
```

IDL DEFINITION

```
[id[id(22), helpstring("method LXEScannerSetSerial")]  
HRESULT LXEScannerSetSerial(LONG port, LONG baud, LONG data, LONG parity,  
LONG stop, IDispatch *pDisp, [out, retval]LONG *pnResult);
```



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult =
    lxe.LXEScannerSetSerial(GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM1,
                            9600,
                            8,
                            1,
                            1,
                            oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSETPOWER**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSETPOWER

LXEScannerSetPower is a method of the lxescanner object. This method changes the value of the power pin value on the specified scanner serial port and saves to registry.

Parameters**port** -

where valid values for port are:

GSW.GSWBrowse.Scanner.LXE.SCANPORT_DISABLED	(0)
GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM1	(1)
GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM2	(2)
GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM3	(3)
GSW.GSWBrowse.Scanner.LXE.SCANPORT_INTERNAL	(4)
GSW.GSWBrowse.Scanner.LXE.SCANPORT_RFID	(5)
GSW.GSWBrowse.Scanner.LXE.SCANPORT_BLUETOOTH	(6)

val -

where valid values for val are:

0	Power Pin is set to ring indicator
1	Power pin is set to 5v.

oHelperObject -

GSWBrowse Helper Object (See [page 234](#))

Return Value

Returns same numeric value as returned by corresponding native LXE API
See *LXE documentation for details*.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerSetPower(port, val, oHelperObject)

IDL DEFINITION

```
[[id(23), helpstring("method LXEScannerSetPower")]
HRESULT LXEScannerSetPower(LONG port, LONG val, IDispatch *pDisp, [out,
retval]LONG *pnResult);
```



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult =
    lxe.LXEScannerSetPower(GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM1,
                           1,
                           oHelperObject);

var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSTRIPLEAD**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSTRIPLEAD

LXEScannerStripLead is a method of the lxescanner object. This method enables or disables the stripping of characters from the beginning of the barcode data.

Parameters**bEnable -**

where valid values for bEnable are:

- 1 = Enable stripping of leading characters
 0 = Disable stripping of leading characters

cnt -

where cnt specifies the number of characters to strip.

oHelperObject -

GSWBrowse Helper Object (See [page 234](#))

Return Value

Returns same numeric value as returned by corresponding native LXE API
See *LXE documentation for details*.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerStripLead(bEnable,Cnt,oHelperObject)

IDL DEFINITION

```
[id(24), helpstring("method LXEScannerStripLead")]
HRESULT LXEScannerStripLead(LONG bEnable, LONG Cnt, IDispatch *pDisp, [out,
retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult = lxe.LXEScannerStripLead(1,3, oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERSTRIPTRAIL**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSTRIPTRAIL

LXEScannerStripTrail is a method of the lxescanner object. This method enables or disables the stripping of characters from the end of the barcode data.

Parameters**bEnable -**

where valid values for bEnable are:
 1 = Enable stripping of trailing characters
 0 = Disable stripping of trailing characters

cnt -

where cnt specifies the number of characters to strip.

oHelperObject -

GSWBrowse Helper Object (See [page 234](#))

Return Value

Returns same numeric value as returned by corresponding native LXE API
 See *LXE documentation for details*.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.**lxescanner.LXEScannerStripTrail(bEnable,Cnt,oHelperObject)**

IDL DEFINITION

```
[id(25), helpstring("method LXEScannerStripTrail")]
HRESULT LXEScannerStripTrail(LONG bEnable, LONG Cnt, IDispatch *pDisp, [out,
retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var nResult = lxe.LXEScannerStripTrail(1,3,oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERCTRLCODEOFF**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.scanner.lxescanner.LXESCANNERCTRLCODEOFF`

LXEScannerCtrlCodeOff is a method of the `lxescanner` object. This method disables/discards processing of ASCII values less than 0x20 when scanner data is processed as keystroke messages.

Parameters**`oHelperObject`**GSWBrowse Helper Object (See [page 234](#))**Return Value**

0 on failure
1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX`window.external.device.scanner.lxescanner.LXEScannerCtrlCodeOff(oHelperObject)`**IDL DEFINITION**

```
[id(26), helpstring("method LXEScannerCtrlCodeOff")]  
HRESULT LXEScannerCtrlCodeOff(IDispatch *pDisp, [out, retval]LONG  
*pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
nResult = Lxe.LXEScannerCtrlCodeOff(oHelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERCTRLCODEON**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERCTRLCODEON

LXEScannerCtrlCodeOn is a method of the lxescanner object. This method enables processing of ASCII values less than 0x20 when scanner data is processed as keystroke messages.

Parameters**oHelperObject**GSWBrowse Helper Object (See [page 234](#))**Return Value**

0 on failure
 2 on success
 3

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerCtrlCodeOn(oHelperObject)

IDL DEFINITION

```
[id(26), helpstring("method LXEScannerCtrlCodeOff")]
HRESULT LXEScannerCtrlCodeOff(IDispatch *pDisp, [out, retval]LONG
*pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
nResult = lxe.LXEScannerCtrlCodeOn(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERPREFIX**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERPREFIX

LXEScannerPrefix is a method of the lxescanner object. This method enables/disables adding a prefix to the barcode data.

Parameters**bEnable**

where valid values for bEnable are:

- 1 = Enable Prefix
 0 = Disable Prefix

strPrefixText

The text to prepend to the barcode data

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

- 0 on failure
 1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerPrefix(bEnable,
strPrefixText, oHelperObject)
```

IDL DEFINITION

```
[id(28), helpstring("method LXEScannerPrefix")]
HRESULT LXEScannerPrefix(LONG bEnable, BSTR PrefixText, IDispatch *pDisp,
[out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult = lxe.LXEScannerPrefix(1,"PP_", oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNERSUFFIX**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSUFFIX

LXEScannerSuffix is a method of the lxescanner object. This method enables/disables adding a suffix to the end barcode data.

Parameters**bEnable**

where valid values for bEnable are:

- 1 = Enable Suffix
 0 = Disable Suffix

strSuffixText

The text to append to the barcode data

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

0 on failure
 1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerSuffix(bEnable,
strSuffixText, oHelperObject)
```

IDL DEFINITION

```
[id(29), helpstring("method LXEScannerSuffix")]
HRESULT LXEScannerSuffix(LONG bEnable, BSTR PrefixText, IDispatch *pDisp,
[out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult = lxe.LXEScannerSuffix(1, "_TT", oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: LXE Device API's

LXESCANNEROPTION**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNEROPTION

LXEScannerOption is a method of the lxescanner object. This method enables/disables symbology options for the internal scanner.

Parameters**nOption -**

where valid values for nOption are:

GSW.GSWBrowse.Scanner.LXE.SCAN_CODE_ENAB

Enable/disable individual symbology

GSW.GSWBrowse.Scanner.LXE.SCAN_AIMID

Enable/disable Symbol aim ID parsing

nCode – used to pass 'code1' parameter

nCodeEx – used to pass the 'code2' parameter.

bEnable -

valid values for bEnable are

1 is Enable symbology

0 is Disable symbology

oHelperObject

GSWBrowse Helper Object (See [page](#) 234)

Return Value

0 on failure

1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerOption(nOption, nCode,
nCodeEx, bEnable, oHelperObject)
```

IDL DEFINITION



```
[id(30), helpstring("method LXEScannerOption")]  
HRESULT LXEScannerOption(LONG nOption, LONG nCode, LONG nCodeEx, LONG  
bEnable, IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
var nResult = lxe.LXEScannerOption(GSW.GSWBrowse.Scanner.LXE.SCAN_CODE_ENAB,  
                                   nCode1,  
                                   nCode2,  
                                   1,  
                                   oHelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERISOPTIONENABLED**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERISOPTIONENABLED

LXEScannerIsOptionEnabled is a method of the lxescanner object. This method enables/disables symbology options for the internal scanner.

Parameters**nOption**

where valid values for nOption are:

GSW.GSWBrowse.Scanner.LXE.SCAN_CODE_ENAB

Determine Enable/disable status of this individual symbology

GSW.GSWBrowse.Scanner.LXE.SCAN_AIMID

Determine Enable/disable status of This individual symbology

nCode – used to pass the 'code1' parameter

nCodeEx – used to pass the 'code2' parameter

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

0 on failure

1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerIsOptionEnabled(nOption,
nCode, nCodeEx, oHelperObject)
```

IDL DEFINITION

```
[id(31), helpstring("method LXEScannerIsOptionEnabled")]
HRESULT LXEScannerIsOptionEnabled(LONG nOption, LONG nCode, LONG nCodeEx,
IDispatch *pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
```



```
var lxe = window.external.device.scanner.lxescanner;  
var nResult =  
lxe.LXEScannerIsOptionEnabled(GSW.GSWBrowse.Scanner.LXE.SCAN_CODE_ENAB,  
                                ncode1,  
                                nCode2,  
                                oHelperObject);  
  
var n0SError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERGETPARAM**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERGETPARAM

LXEScannerGetParam is a method of the lxescanner object. This method gets the value of a hardware specific parameter directly from the scanner hardware. The parameter number is passed in parm and the current value is returned in the Int1 member of the GSW Helper Object.

Parameters**param**

Parameter number from scanner datasheet.

val

Used to pass the 'val' parameter to the function.

flags

valid values for flags are:

0 = Error checking is not performed
 GSW.GSWBrowse.Scanner.LXE.FLAG_SYMBOL = 1
 for Symbol Scanners
 GSW.GSWBrowse.Scanner.LXE.FLAG_IMAGER = 2
 for Intermec imagers

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Parameter Passing

val - returned in Int1 member of GSW Helper
 The value of the hardware specific parameter

Return Value

0 on failure
 1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerGetParam(param, flags, val, oHelperObject)
```

IDL DEFINITION



```
[id(32), helpstring("method LXEScannerGetParam")]  
HRESULT LXEScannerGetParam(LONG param, LONG flags, IDispatch *pDisp, [out,  
retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
var nResult =  
    lxe.LXEScannerGetParam(nParam, GSW.GSWBrowse.Scanner.LXE.FLAG_SYMBOL,  
                           oHelperObject);  
  
var nVal = oHelperObject.Int1;  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSETPARAM**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSETPARAM

LXEScannerSetParam is a method of the lxescanner object. This method sets the value of a hardware specific parameter directly from the scanner hardware. The parameter number is passed in parm and the current value is returned in the Int1 member of the GSW Helper Object.

Parameters**param**

Parameter number from scanner datasheet.

val

Used to pass the 'val' parameter to the function.

flags

valid values for flags are:

0 = Error checking is not performed
 GSW.GSWBrowse.Scanner.LXE.FLAG_SYMBOL = 1
 for Symbol Scanners
 GSW.GSWBrowse.Scanner.LXE.FLAG_IMAGER = 2
 for Intermec imagers

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

0 on failure
 1 on success

val - returned in Int1 member of GSW Helper
 The value of the hardware specific parameter

OS Error

OS Error (if any) is available in the
[LastError](#) member of the GSW Helper Object
 after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerSetParam(param, val, flags, oHelperObject)
```

IDL DEFINITION

```
[id(33), helpstring("method LXEScannerSetParam")]
```



```
HRESULT LXEScannerSetParam(LONG param, LONG val, LONG flags, IDispatch  
*pDisp, [out, retval]LONG *pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
var nResult = lxe.LXEScannerSetParam(nParam, nValue,  
                                     GSW.GSWBrowse.Scanner.LXE.FLAG_SYMBOL, oHelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSYMSTRIPLEAD**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

`window.external.device.scanner.lxescanner.LXESCANNERSYMSTRIPLEAD`

LXEScannerSymStripLead is a method of the lxescanner object. This method enables or disables the stripping of characters from the beginning of the barcode data for specified symbologies.

Parameters**symb**

where valid values for symb are listed in the LXE document Symbology Descriptors¹⁰.

bEnable

where valid values for bEnable are:
1 = Enable stripping of leading characters
0 = Disable stripping of leading characters

cnt

where cnt specifies the number of characters to strip. Max = 99

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerSymStripLead(symb,  
bEnable,Cnt,oHelperObject)
```

IDL DEFINITION

```
[id(34), helpstring("method LXEScannerSymStripLead")]  
HRESULT LXEScannerSymStripLead(BSTR symb, LONG bEnable, LONG Cnt, IDispatch  
*pDisp, [out, retval]LONG *pnResult);
```

¹⁰ Symbologies constants are also available on page 222



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult =
    lxe.LXEScannerSymStripLead(GSW.GSWBrowse.Scanner.LXE.SYMB_Code39, 1, 2,
                               oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSYMSTRIPTRAIL**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSYMSTRIPTRAIL

LXEScannerSymStripTrail is a method of the lxescanner object. This method enables or disables the stripping of characters from the end of the barcode data for the specified symbology.

Parameters**symb**

where valid values for symb are listed in the LXE document Symbology Descriptors¹¹.

bEnable

where valid values for bEnable are:
1 = Enable stripping of trailing characters
0 = Disable stripping of trailing characters

cnt

where cnt specifies the number of characters to strip. Max = 99

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerSymStripTrail(symb,
bEnable,Cnt,oHelperObject)
```

IDL DEFINITION

```
[id(35), helpstring("method LXEScannerSymStripTrail")]
HRESULT LXEScannerSymStripTrail(BSTR symb, LONG bEnable, LONG Cnt, IDispatch
*pDisp, [out, retval]LONG *pnResult);
```

¹¹ Symbologies constants are also available on page 222



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult =
    lxe.LXEScannerSymStripTrail(GSW.GSWBrowse.Scanner.LXE.SYMB_Code39,
                                1,
                                2,
                                oHelperObject);
var n0SError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSYMPREFIX**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSYMPREFIX

LXEScannerSymPrefix is a method of the lxescanner object. This method enables/disables adding a prefix to the barcode data for the specified symbology.

Parameters**symb**

where valid values for symb are listed in the LXE Document Symbology Descriptors¹².

bEnable

where valid values for bEnable are:

- 1 = Enable Suffix
 0 = Disable Suffix

strPrefix

The text to prepend to the barcode data

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

- 0 on failure
 1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

```
window.external.device.scanner.lxescanner.LXEScannerSymPrefix(symb, bEnable, strPrefix, oHelperObject)
```

IDL DEFINITION

```
[id(36), helpstring("method LXEScannerSymPrefix")]
HRESULT LXEScannerSymPrefix(BSTR symb, LONG bEnable, BSTR Prefix, IDispatch *pDisp, [out, retval]LONG *pnResult);
```

¹² Symbologies constants are also available on page 222



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult = lxe.LXEScannerSymPrefix(GSW.GSWBrowse.Scanner.LXE.SYMB_Code39,
                                     1,
                                     "PP_",
                                     oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSYMSUFFIX**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSYMSUFFIX

LXEScannerSymSuffix is a method of the lxescanner object. This method enables/disables adding a suffix to the end barcode data for the specified symbology.

Parameters**symb**

where valid values for symb are listed in the LXE document Symbology Descriptors¹³.

bEnable

where valid values for bEnable are:

- 1 = Enable Suffix
 0 = Disable Suffix

strSuffix

The text to prepend to the barcode data

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

- 0 on failure
 1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerSymSuffix(symb, bEnable, strSuffix, oHelperObject)

IDL DEFINITION

```
[id(37), helpstring("method LXEScannerSymSuffix")]
HRESULT LXEScannerSymSuffix(BSTR symb, LONG bEnable, BSTR Suffix, IDispatch
*pDisp, [out, retval]LONG *pnResult);
```

¹³ Symbologies constants are also available on page 222



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
var nResult = lxe.LXEScannerSymSuffix(GSW.GSWBrowse.Scanner.LXE.SYMB_Code39,  
                                     1,  
                                     "_TT",  
                                     HelperObject);  
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSYMENABLE**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSYMENABLE

LXEScannerSymEnable is a method of the lxescanner object. Enables or disables a specified symbology for the attached scanner.

Parameters**symb**

where valid values are listed in the LXE document Symbology Descriptors¹⁴.

bEnable

where valid values for bEnable are:

- 1 = Enable Symbology
- 0 = Disable Symbology

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

Returns same numeric value as returned by corresponding native LXE API
See LXE documentation for details.

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerSymEnable(symb, bEnable, oHelperObject)

IDL DEFINITION

```
[id(38), helpstring("method LXEScannerSymEnable")]
HRESULT LXEScannerSymEnable(BSTR symb, LONG bEnable, IDispatch *pDisp, [out,
retval]LONG *pnResult);
```

¹⁴ Symbologies constants are also available on page 222



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult = lxe.LXEScannerSymEnable(GSW.GSWBrowse.Scanner.LXE.SYMB_Code39,
                                     1,
                                     oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSYMISENABLED**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSYMISENABLED

LXEScannerSymIsEnabled is a method of the lxescanner object. This method returns the current enable/disable state for the specified symbology.

Parameters**symb**

where valid values are listed in the LXE Document Symbology Descriptors¹⁵.

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Parameter Passing

Int1 - After the call the value of 'enable' is placed in the Int1 member of the GSW Helper Object.

Return Value

0 on failure
1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerSymIsEnabled(symb, oHelperObject)

IDL DEFINITION

```
[id(31), helpstring("method LXEScannerIsOptionEnabled")]
HRESULT LXEScannerIsOptionEnabled(LONG nOption, LONG nCode, LONG nCodeEx,
IDispatch *pDisp, [out, retval]LONG *pnResult);
```

¹⁵ Symbologies constants are also available on page 222



EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();  
var lxe = window.external.device.scanner.lxescanner;  
var nResult =  
    lxe.LXEScannerSymIsEnabled(GSW.GSWBrowse.Scanner.LXE.SYMB_Code39,  
                                oHelperObject);  
var nOSError = oHelperObject.LastError;  
var bEnabled = oHelperObject.Int1;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSYMGETCONFIG**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSYMGETCONFIG

LXEScannerSymGetConfig is a method of the lxescanner object. This method returns the current parameters for the specified symbology.

Parameters**symb**

where valid values are listed in the LXE document *Symbology Descriptors*¹⁶.

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Parameter Passing

All the following are returned in the GSW Helper Object

name[32]	Member <i>String1</i>
codenab	Member <i>Int1</i>
minlen	Member <i>Int2</i>
maxlen	Member <i>Int3</i>
striplead	Member <i>Int4</i>
leadval	Member <i>Int5</i>
striptrail	Member <i>Int6</i>
trailval	Member <i>Int7</i>
stripid	Member <i>Int8</i>
useprefix	Member <i>Int9</i>
usesuffix	Member <i>Int10</i>
Prefix[64]	Member <i>String2</i>
Suffix[64]	Member <i>String3</i>

Return Value

0 on failure
1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerSymGetConfig(symb, oHelperObject)

¹⁶ Symbologies constants are also available on page 222

IDL DEFINITION

```
[id(40), helpstring("method LXEScannerSymGetConfig")]
HRESULT LXEScannerSymGetConfig(BSTR symb, IDispatch *pDisp, [out,
retval]LONG *pnResult);
```

EXAMPLE - Segment

```
/*
typedef struct {
    WCHAR name[32];           //String1
    DWORD codenab;            //Int1
    DWORD minlen;             //Int2
    DWORD maxlen;            //Int3
    DWORD striplead;          //Int4
    DWORD leadval;            //Int5
    DWORD striptrail;         //Int6
    DWORD trailval;           //Int7
    DWORD stripid;            //Int8
    DWORD useprefix;          //Int9
    DWORD usesuffix;          //Int10
    WCHAR prefix[64];         //String2
    WCHAR suffix[64];         //String3
} SCAN_SYMBLOGY;
*/

var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult =
    lxe.LXEScannerSymGetConfig(GSW.GSWBrowse.Scanner.LXE.SYMB_Code39,
                                oHelperObject);

var nOSError = oHelperObject.LastError;
var codenab = oHelperObject.Int1;
var minlen = oHelperObject.Int2;
var maxlen = oHelperObject.Int3;
var striplead = oHelperObject.Int4;
var leadval = oHelperObject.Int5;
var striptrail = oHelperObject.Int6;
var trailval = oHelperObject.Int7;
var stripid = oHelperObject.Int8;
var useprefix = oHelperObject.Int9;
var usesuffix = oHelperObject.Int10;
var prefix = oHelperObject.String2;
var suffix = oHelperObject.String3;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSYMSETCONFIG**SYNTAX**

- ☐ Meta Tag
☐ Navigational
☒ Windows External

SCOPE

- ☒ DEVICE
☐ BROWSER -INSTANCE
☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSYMSETCONFIG

LXEScannerSymSetConfig is a method of the lxescanner object. This method sets/configures the parameters for the specified symbology.

Parameters**oHelperObject**GSWBrowse Helper Object (See [page 234](#))**Parameter Passing**

All the following are returned in the GSW Helper Object

Note: The symbology is placed in string1

name[32] Member *String1*
codenab Member *Int1*
minlen Member *Int2*
maxlen Member *Int3*
striplead Member *Int4*
leadval Member *Int5*
striptrail Member *Int6*
trailval Member *Int7*
stripid Member *Int8*
useprefix Member *Int9*
usesuffix Member *Int10*
Prefix[64] Member *String2*
Suffix[64] Member *String3*

Return Value

0 on failure
1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerSymSetConfig(oHelperObject)

IDL DEFINITION

```
[id(41), helpstring("method LXEScannerSymSetConfig")]
HRESULT LXEScannerSymSetConfig(IDispatch *pDisp, [out, retval]LONG
*pnResult);
```



EXAMPLE - Segment

```
/*
typedef struct {
    WCHAR name[32];           //String1
    DWORD codenab;           //Int1
    DWORD minlen;            //Int2
    DWORD maxlen;            //Int3
    DWORD striplead;          //Int4
    DWORD leadval;           //Int5
    DWORD striptrail;         //Int6
    DWORD trailval;          //Int7
    DWORD stripid;           //Int8
    DWORD useprefix;          //Int9
    DWORD usesuffix;          //Int10
    WCHAR prefix[64];         //String2
    WCHAR suffix[64];         //String3
} SCAN_SYMBLOGY;
*/
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult =
    lxe.LXEScannerSymGetConfig(GSW.GSWBrowse.Scanner.LXE.SYMB_Code39,
                              oHelperObject);
var nOSError = oHelperObject.LastError;
var striplead = 1;
oHelperObject.Int4 = striplead;
var leadval = 3;
var oHelperObject.Int5 = leadval;
var nResult = lxe.LXEScannerSymSetConfig(oHelperObject);
var nOSError = oHelperObject.LastError;
```



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.



Category: LXE Device API's

LXESCANNERSYREMOVE**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.device.scanner.lxescanner.LXESCANNERSYMREMOVE

LXEScannerSymRemove is a method of the lxescanner object. This method removes all the parameters for the specified symbology.

Parameters**symb**

where valid values are listed in the LXE document Symbology Descriptors¹⁷.

oHelperObject

GSWBrowse Helper Object (See [page 234](#))

Return Value

0 on failure

1 on success

OS Error

OS Error (if any) is available in the [LastError](#) member of the GSW Helper Object after the call returns.

EXTERNAL SYNTAX

window.external.device.scanner.lxescanner.LXEScannerSymRemove(symb, oHelperObject)

IDL DEFINITION

```
[id(42), helpstring("method LXEScannerSymRemove")]
HRESULT LXEScannerSymRemove(BSTR symb, IDispatch *pDisp, [out, retval]LONG
*pnResult);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
var lxe = window.external.device.scanner.lxescanner;
var nResult = lxe.LXEScannerSymRemove(GSW.GSWBrowse.Scanner.LXE.SYMB_Code39,
                                         oHelperObject);
var nOSError = oHelperObject.LastError;
```

¹⁷ Symbologies constants are also available on page 222



- LXE Only – See LXE CE API Programming Guide for details, updates and changes.

Category: Device API's

GSW HELPER OBJECT

SYNTAX

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

GSW HELPER OBJECT

Georgia SoftWorks is able to provide specialized interfaces to device API's further extending the programming power of GSWBrowse. With these specialized interfaces the developer gains access to extra device features while maintaining a consistent development style and enjoying the benefits of GSWBrowse.

A technical hurdle is that JavaScript is unable to pass parameters by reference to methods of COM objects. Native device API's (provided by the device vendor) typically use pointers and pass parameters by reference, which makes it impossible to create window.external methods with the same synopsis.

To work around this technical problem GSWBrowse has created a "GSW Helper" COM object which provides functionality analogous to passing parameters by reference with the native device API COM object.

GSWBrowse Helper Object provides an area that is used for data exchange with the COM object. The Helper Object is passed to the COM object rather than individual data. After the call any data passed back to the client is retrieved from the Helper Object.

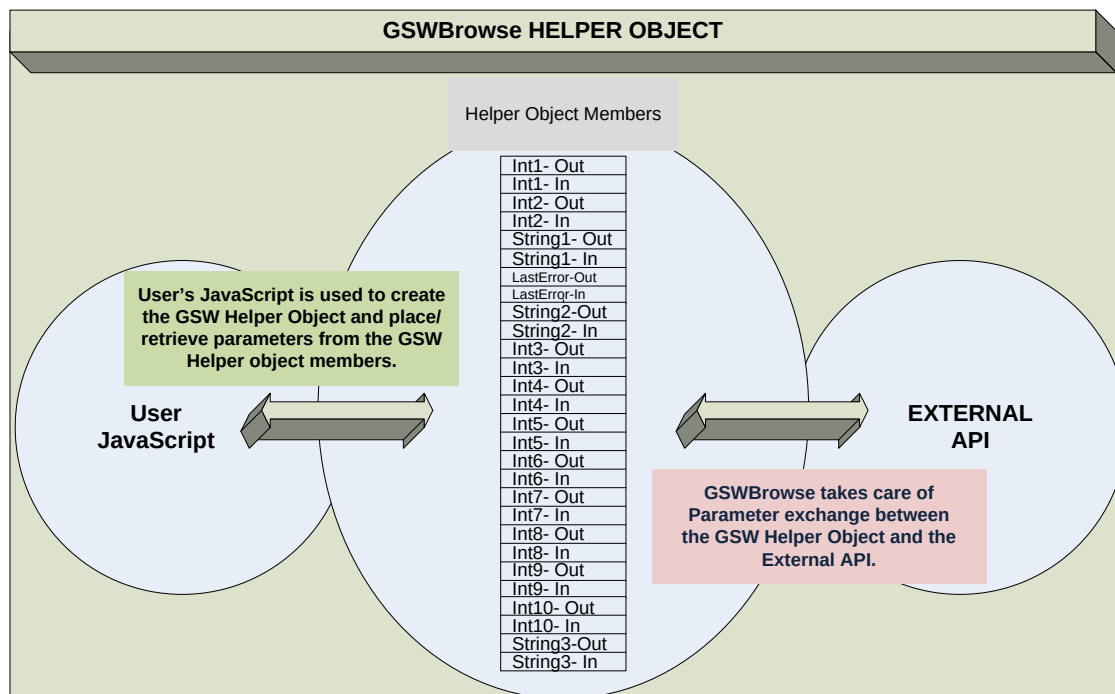


Figure 58: GSWBrowse Helper Object - Overview

Following is a simple example to depict the typical steps in how to use the GSW Helper Object.

1. Create The GSW Helper Object
2. Load parameters (if any) in the GSW Helper Object.
3. Initiate External API call.
4. Retrieve any Return Values from GSW Helper Object.

1. Create GSW Helper Object

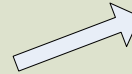
CreateParameterBlock

creates and returns the GSW Helper Object with room for ten integers, three strings and error status

A typical call would look something like:

```
var oHelperObject = window.external.interface.CreateParameterBlock();
```

Upon successful return, oHelperObject would contain an object with members as shown to the right.



Int1- Out
Int1- In
Int2- Out
Int2- In
String1- Out
String1- In
LastError-Out
LastError-In
String2-Out
String2- In
Int3- Out
Int3- In
Int4- Out
Int4- In
Int5- Out
Int5- In
Int6- Out
Int6- In
Int7- Out
Int7- In
Int8- Out
Int8- In
Int9- Out
Int9- In
Int10- Out
Int10- In
String3-Out
String3- In

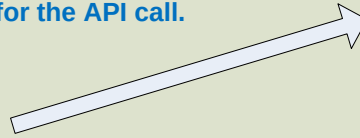
Figure 59: GSW Helper Object - Overview - Create Parameter Block

2. Load values(if any) into object members for the API call.

Example:

```
oHelperObject.Int1 = 100;
```

This loads the numeric value 100 into the Int1 member of the GSW Helper Object



Int1 - Out
100 Int1- In
Int2- Out
Int2- In
String1- Out
String1- In
LastError-Out
LastError-In
String2-Out
String2- In
Int3- Out
Int3- In
Int4- Out
Int4- In
Int5- Out
Int5- In
Int6- Out
Int6- In
Int7- Out
Int7- In
Int8- Out
Int8- In
Int9- Out
Int9- In
Int10- Out
Int10- In
String3-Out
String3- In

3. Make External API call.

Example:

```
var lxe = windows.external.device.scanner.lxescanner;  
nResult = lxe.LXEScannerGetData(oHelperObject);  
nOSError = oHelperObject.LastError;
```

Figure 60: GSW Helper Object - Overview - Load Values



4. Upon successful completion, get data from GSW Helper Object.

```
var nScannedDataLength = oHelperObject.Int1;  
var strScannedData = oHelperObject.String1;
```

Lets say in this example the Scanned Data was:

Georgia SoftWorks

Then 17 (length of Georgia SoftWorks) would be in Int1

And the String "Georgia SoftWorks" would be in String1

17	Int1- Out
	Int1- In
	Int2- Out
	Int2- In
Georgia SoftWorks	String1- In
	LastError- Out
	LastError- In
	String2- Out
	String2- In
	Int3- Out
	Int3- In
	Int4- Out
	Int4- In
	Int5- Out
	Int5- In
	Int6- Out
	Int6- In
	Int7- Out
	Int7- In
	Int8- Out
	Int8- In
	Int9- Out
	Int9- In
	Int10- Out
	Int10- In
	String3- Out
	String3- In

Figure 61: GSWBrowse Helper Object - Overview - Retrieve Data



Category: Interface API

CREATEPARAMETERBLOCK**SYNTAX**

- ☐ Meta Tag
- ☐ Navigational
- ☒ Windows External

SCOPE

- ☒ DEVICE
- ☐ BROWSER -INSTANCE
- ☐ BROWSER -GLOBAL

window.external.interface.CREATEPARAMETERBLOCK

CreateParameterBlock is a method of the interface object. This method creates and returns the GSW Helper Object used for parameter exchange with external API's and JavaScript.

Parameters None

Return Value GSWBrowse Helper Object

EXTERNAL SYNTAX

window.external.**interface.CreateParameterBlock()**

IDL DEFINITION

```
[[id(1), helpstring("method CreateParameterBlock")]
HRESULT CreateParameterBlock([out, retval]IDispatch* *params);
```

EXAMPLE - Segment

```
var oHelperObject = window.external.interface.CreateParameterBlock();
```

EXAMPLE - CreateParameterBlock

*Duplicating one of the LXE examples, except this one has used **red** color for the font on the lines that use CreateParameterBlock or the GSW Browse Helper Object Members.*

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<meta http-equiv="gswbrowse_ScannerNavigate" content="javascript:OnScan('%s', '%s', '%s', '%s', '%s');">
<script src="../scripts/GSWBrowse.js"></script>
<script language=javascript>
function OnScan(a,b,c,d,e)
{
    document.getElementById("scandiv").innerHTML= a + " " + b + " " + c + " " + d + " " + e;
}
Display everything that
function ShowObject(arg1, result)
{
    var property, properties="";
    properties += ("Int1: " + arg1.Int1 + "\n");
```



```
properties += ("Int2: " + arg1.Int2 + "\n");
properties += ("Int3: " + arg1.Int3 + "\n");
properties += ("Int4: " + arg1.Int4 + "\n");
properties += ("Int5: " + arg1.Int5 + "\n");
properties += ("Int6: " + arg1.Int6 + "\n");
properties += ("Int7: " + arg1.Int7 + "\n");
properties += ("Int8: " + arg1.Int8 + "\n");
properties += ("Int9: " + arg1.Int9 + "\n");
properties += ("Int10: " + arg1.Int10 + "\n");
properties += ("String1: " + arg1.String1 + "\n");
properties += ("String2: " + arg1.String2 + "\n");
properties += ("String3: " + arg1.String3 + "\n");
properties += ("LastError: " + arg1.LastError + "\n");
properties += ("Result of op: " + result);

alert(properties);
}

function Vibrate()
{
    window.external.device.vibrate(1000);
}

function ConfigureScanner()
{
    var lxe=window.external.device.scanner.lxescanner;
    var params = window.external.interface.CreateParameterBlock();
    var result;
    var error;

    result = lxe.LXEScannerPrefix(1, 'PP_', params);
    error = params.LastError;
    document.Commander.info.value += '(' + result + ',' + error + '),';

    result = lxe.LXEScannerSuffix(1, '_QQ', params);
    error = params.LastError;
    document.Commander.info.value += '(' + result + ',' + error + '),';

    result = lxe.LXEScannerStripLead(1, 2, params);
    error = params.LastError;
    document.Commander.info.value += '(' + result + ',' + error + '),';

    result = lxe.LXEScannerStripTrail(1, 2, params);
    error = params.LastError;
    document.Commander.info.value += '(' + result + ',' + error + '),';
}
```

This function shows how to call LXEScannerSymGetConfig

```
function ShowScannerConfig()
{
    var lxe=window.external.device.scanner.lxescanner;
    var params = window.external.interface.CreateParameterBlock();
    var result;
    var error;

    /*

typedef struct {
    WCHAR name[32];           //String1
    DWORD codenab;            //Int1
    DWORD minlen;             //Int2
    DWORD maxlen;             //Int3
    DWORD striplead;          //Int4
    DWORD leadval;            //Int5
    DWORD striptrail;         //Int6
    DWORD trailval;           //Int7
    DWORD stripid;            //Int8
```



```
        DWORD useprefix;        //Int9
        DWORD usesuffix;        //Int10
        WCHAR prefix[64];        //String2
        WCHAR suffix[64];        //String3
    } SCAN_SYMBOLGY;
    */
    Pass it pointer to symbology, and the pointer to the parameter block(params).
    result = lxe.LXEScannerSymGetConfig(GSW.GSWBrowse.Scanner.LXE.SYMB_ALL, params);
    ShowObject(params, result);
}
</script>
</head>

<body>

<center>
<table border=5 bordercolor=black cellspacing=8>
<form name="Commander">
<th>
The Command Center
</th>
<tr align="center"><td>
        <input type="text" name="info" maxlength=10 size=24>
</td></tr>
<tr align="center"><td>
        <input type="button" value=" Vibrate " onclick="Vibrate()">
        <input type="button" value=" Configure Scanner " onclick="ConfigureScanner()">
        <input type="button" value=" Show Config " onclick="ShowScannerConfig()">
</td></tr>
</form>
</table>
Scan: <div id="scandiv">junk</div>
</body>
</html>
```



IDL DEFINITION

```
[propget, id(1), helpstring("property Int1")] HRESULT Int1([out, retval] long *pVal);
[propput, id(1), helpstring("property Int1")] HRESULT Int1([in] long newVal);
[propget, id(2), helpstring("property Int2")] HRESULT Int2([out, retval] long *pVal);
[propput, id(2), helpstring("property Int2")] HRESULT Int2([in] long newVal);
[propget, id(3), helpstring("property String1")] HRESULT String1([out, retval] BSTR *pVal);
[propput, id(3), helpstring("property String1")] HRESULT String1([in] BSTR newVal);
[propget, id(4), helpstring("property LastError")] HRESULT LastError([out, retval] long *pVal);
[propput, id(4), helpstring("property LastError")] HRESULT LastError([in] long newVal);
[propget, id(5), helpstring("property String2")] HRESULT String2([out, retval] BSTR *pVal);
[propput, id(5), helpstring("property String2")] HRESULT String2([in] BSTR newVal);
[propget, id(6), helpstring("property Int3")] HRESULT Int3([out, retval] long *pVal);
[propput, id(6), helpstring("property Int3")] HRESULT Int3([in] long newVal);
[propget, id(7), helpstring("property Int4")] HRESULT Int4([out, retval] long *pVal);
[propput, id(7), helpstring("property Int4")] HRESULT Int4([in] long newVal);
[propget, id(8), helpstring("property Int5")] HRESULT Int5([out, retval] long *pVal);
[propput, id(8), helpstring("property Int5")] HRESULT Int5([in] long newVal);
[propget, id(9), helpstring("property Int6")] HRESULT Int6([out, retval] long *pVal);
[propput, id(9), helpstring("property Int6")] HRESULT Int6([in] long newVal);
[propget, id(10), helpstring("property Int7")] HRESULT Int7([out, retval] long *pVal);
[propput, id(10), helpstring("property Int7")] HRESULT Int7([in] long newVal);
[propget, id(11), helpstring("property Int8")] HRESULT Int8([out, retval] long *pVal);
[propput, id(11), helpstring("property Int8")] HRESULT Int8([in] long newVal);
[propget, id(12), helpstring("property Int9")] HRESULT Int9([out, retval] long *pVal);
[propput, id(12), helpstring("property Int9")] HRESULT Int9([in] long newVal);
[propget, id(13), helpstring("property Int10")] HRESULT Int10([out, retval] long *pVal);
[propput, id(13), helpstring("property Int10")] HRESULT Int10([in] long newVal);
[propget, id(14), helpstring("property String3")] HRESULT String3([out, retval] BSTR *pVal);
[propput, id(14), helpstring("property String3")] HRESULT String3([in] BSTR newVal);
```



Selected Samples from CD

Several running examples are included on the CD and in the download.

BATT_NAV.HTM

// On CD: Folder GSWebBrowse/meta/batt_nav.htm

```
<html>

<head>

<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswebbrowse_BatteryNavigate"
CONTENT="Javascript:BatteryTestfun('%s','%s','%s','%s');">
<META HTTP-EQUIV="gswebbrowse_BatteryTimerInterval" CONTENT="2000">
<META HTTP-EQUIV="gswebbrowse_TimerNavigate" CONTENT="Javascript:TimerTestfun('%s');">
<META HTTP-EQUIV="gswebbrowse_TimerInterval" CONTENT="1000">

<script language=javascript>
function BatteryTestfun(a,b,c,d)
{
    document.getElementById("batterydiv").innerHTML= a + " " + b + " " + c + " " +d;
}
function TimerTestfun(a)
{
    document.getElementById("battery_statustext").innerHTML=
window.external.device.battery.statustext;
    document.getElementById("battery_status").innerHTML=
window.external.device.battery.status;
    document.getElementById("battery_level").innerHTML=
window.external.device.battery.level;
}
</script>

</head>

<body>
<p>The BatteryNavigate tag assumes your device has a battery. No
navigation or script call shall occur if a battery is not found.</p>
Current battery state: <div id="batterydiv">junk</div>
<p>External battery properties</p>
battery.statustext: <div id="battery_statustext">junk</div>
battery.status: <div id="battery_status">junk</div>
battery.level: <div id="battery_level">junk</div>
<p><a href="default.htm">Return to main menu</a></p>
</body>

</html>
```



DATETIME.HTM

// On CD: Folder GSWBrowse/meta/datetime.htm

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswbrowse_SetDate" CONTENT="09/13/2011">
<META HTTP-EQUIV="gswbrowse_SetTime" CONTENT="19:20">
</head>

<body>
You should see date and time set to 09/13/2011 19:20 (GMT)
<p><a href="default.htm">Return to main menu</a></p>
</body>
```



EXTERNAL.HTM

// On CD: Folder GSWBrowse/meta/external.htm

```
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="gswwbrowse_OnKeyVK_F2" CONTENT="Javascript:OnVK_F2();">
<script type="text/javascript">
function OnVK_F2()
{
    alert(window.external.app.name + " " + window.external.app.version +
        " " + window.external.app.buildid);
    //alert("Test");
}
</script>
</head>

<body onload="javascript:fetchvalues();">
<form id="form1" name="form1">
Test of GSWBrowse Object Model<br>
window.external.device.serialnumber<input type="text" name="serialnumber" size="40"><br>
window.external.device.ipaddress<input type="text" name="ipaddress" size="40"><br>
window.external.device.batterystatus<input type="text" name="batterystatus" size="40"><br>
window.external.device.battery.statustext<input type="text" name="statustext" size="40"><br>
window.external.device.battery.status<input type="text" name="status" size="40"><br>
window.external.device.battery.level<input type="text" name="level" size="40"><br>
window.external.device.deviceid<input type="text" name="deviceid" size="40"><br>
window.external.device.macaddress<input type="text" name="macaddress" size="40"><br>
window.external.device.platformid<input type="text" name="platformid" size="40"><br>
window.external.device.presetid<input type="text" name="presetid" size="40"><br>
window.external.device.uuid<input type="text" name="uuid" size="40"><br>
window.external.device.apiversion<input type="text" name="apiversion" size="50"><br>
window.external.device.serialnumber<input type="text" name="serialnumber" size="50"><br>

window.external.device.NetworkAdapterName<input type="text" name="adapter" size="40"><br>
window.external.device.rf.strength<input type="text" name="rfstrength" size="40"><br>
window.external.device.rf.status<input type="text" name="rfstatus" size="40"><br>
window.external.app.buildid<input type="text" name="buildid" size="40"><br>
window.external.app.name<input type="text" name="name" size="40"><br>
window.external.app.activesessionindex<input type="text" name="active" size="40"><br>
window.external.app.version<input type="text" name="version" size="40"><br>
window.external.app.copyright<input type="text" name="copyright" size="40"><br>
window.external.app.usertext(15)<input type="text" name="usertext15" size="40"><br>
idx=window.external.ThisSessionIndex<input type="text" name="thisidx" size="40"><br>
window.external.app.session(idx).host<input type="text" name="host" size="40"><br>
window.external.app.session(idx).port<input type="text" name="port" size="40"><br>
window.external.app.session(idx).type<input type="text" name="type" size="40"><br>
window.external.app.session(idx).path<input type="text" name="path" size="40"><br>
window.external.app.session(idx).homepage<input type="text" name="homepage" size="60"><br>
</form>

<script language=javascript>
function fetchvalues()
{
    var device = window.external.device;
    var rf = window.external.device.rf;
    var app = window.external.app;

    app.usertext(15) = "text 15";
    document.form1.serialnumber.value = device.serialnumber;
    document.form1.ipaddress.value = device.ipaddress;
    document.form1.batterystatus.value = device.batterystatus;
    document.form1.statustext.value = device.battery.statustext;
    document.form1.status.value = device.battery.status;
    document.form1.level.value = device.battery.level;
    document.form1.deviceid.value = device.deviceid;
    document.form1.macaddress.value = device.macaddress;
    document.form1.platformid.value = device.platformid;
```



```
document.form1.presetid.value = device.presetid;
document.form1.serialnumber.value = device.serialnumber;

document.form1.uuid.value = device.uuid;
document.form1.apiversion.value = device.APIVersion;
document.form1.adapter.value = device.NetworkAdapterName;
document.form1.rfstrength.value = rf.strength;
document.form1.rfstatus.value = rf.status;
document.form1.buildid.value = app.buildid;
document.form1.name.value = app.name;
document.form1.active.value = app.activesessionindex;
document.form1.version.value = app.version;
document.form1.version.value = app.copyright;
document.form1.usertext15.value = app.usertext(15);

var idx = window.external.ThisSessionIndex;
document.form1.thisidx.value = idx;
document.form1.host.value = app.session(idx).host;
document.form1.port.value = app.session(idx).port;
document.form1.type.value = app.session(idx).type;
document.form1.path.value = app.session(idx).path;
document.form1.homepage.value = app.session(idx).homepage;

}
</script>

<p><a href="default.htm">Return to main menu</a></p>
</body>
```



LXE_HELPER_OBJECT.HTM

JavaScript you cannot get more than one return value from a function.

```
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<meta http-equiv="gswbrowse_ScannerNavigate" content="javascript:OnScan('%s', '%s', '%s', '%s', '%s');">
<script src="../scripts/GSWBrowse.js"></script>
<script language=javascript>
function OnScan(a,b,c,d,e)
{
    document.getElementById("scandiv").innerHTML= a + " " + b + " " + c + " " + d + " " + e;
}
Display everything that
function ShowObject(arg1, result)
{
    var property, properties="";

    properties += ("Int1: " + arg1.Int1 + "\n");
    properties += ("Int2: " + arg1.Int2 + "\n");
    properties += ("Int3: " + arg1.Int3 + "\n");
    properties += ("Int4: " + arg1.Int4 + "\n");
    properties += ("Int5: " + arg1.Int5 + "\n");
    properties += ("Int6: " + arg1.Int6 + "\n");
    properties += ("Int7: " + arg1.Int7 + "\n");
    properties += ("Int8: " + arg1.Int8 + "\n");
    properties += ("Int9: " + arg1.Int9 + "\n");
    properties += ("Int10: " + arg1.Int10 + "\n");
    properties += ("String1: " + arg1.String1 + "\n");
    properties += ("String2: " + arg1.String2 + "\n");
    properties += ("String3: " + arg1.String3 + "\n");
    properties += ("LastError: " + arg1.LastError + "\n");
    properties += ("Result of op: " + result);

    alert(properties);
}

function Vibrate()
{
    window.external.device.vibrate(1000);
}

function ConfigureScanner()
{
    var lxe=window.external.device.scanner.lxescanner;
    var params = window.external.interface.CreateParameterBlock();
    var result;
    var error;

    result = lxe.LXEScannerPrefix(1, 'PP_', params);
    error = params.LastError;
    document.Commander.info.value += '(' + result + ',' + error + '),';

    result = lxe.LXEScannerSuffix(1, '_QQ', params);
    error = params.LastError;
    document.Commander.info.value += '(' + result + ',' + error + '),';

    result = lxe.LXEScannerStripLead(1, 2, params);
    error = params.LastError;
    document.Commander.info.value += '(' + result + ',' + error + '),';

    result = lxe.LXEScannerStripTrail(1, 2, params);
    error = params.LastError;
    document.Commander.info.value += '(' + result + ',' + error + '),';
}
```



This function shows how to call LXEScannerSymGetConfig

```
function ShowScannerConfig()
{
    var lxe=window.external.device.scanner.lxescanner;
    var params = window.external.interface.CreateParameterBlock();
    var result;
    var error;

    /*

typedef struct {
    WCHAR name[32];           //String1
    DWORD codenab;           //Int1
    DWORD minlen;            //Int2
    DWORD maxlen;           //Int3
    DWORD striplead;         //Int4
    DWORD leadval;          //Int5
    DWORD striptrail;        //Int6
    DWORD trailval;         //Int7
    DWORD stripid;          //Int8
    DWORD useprefix;         //Int9
    DWORD usesuffix;        //Int10
    WCHAR prefix[64];        //String2
    WCHAR suffix[64];        //String3
} SCAN_SYBBOLOGY;
*/
    Pass it pointer to symbology, and the pointer to the parameter block(params).
    result = lxe.LXEScannerSymGetConfig(GSW.GSWBrowse.Scanner.LXE.SYMB_ALL, params);
    ShowObject(params, result);
}
</script>
</head>

<body>

<center>
<table border=5 bordercolor=black cellspacing=8>
<form name="Commander">
<th>
The Command Center
</th>
<tr align="center"><td>



</td></tr>
</form>
</table>
Scan: <div id="scandiv">junk</div>
</body>
</html>
```



Developer Objects and Constants Reference

GSW Programming Objects – Interface Description Language

GSWBROWSE:IDISPATCH

```
interface IAtlBrCon : IDispatch
{
    [propget, id(DISPID_APP), helpstring("property app")]
        HRESULT App([out, retval] IDispatch* *pVal);
    [propget, id(DISPID_DEVICE), helpstring("property device")]
        HRESULT Device([out, retval] IDispatch* *pVal);
    [propget, id(DISPID_THIS_SESSION_INDEX), helpstring("property ThisSessionIndex")]
        HRESULT ThisSessionIndex([out, retval] long *pVal);
    [propget, id(DISPID_INTERFACE), helpstring("property interface")]
        HRESULT Interface([out, retval] IDispatch* *pVal);
};
```

IEXTERNALAPP:IDISPATCH

```
interface IExternalApp : IDispatch
{
    [propget, id(1), helpstring("property Version")]
        HRESULT Version([out, retval] BSTR *pVal);
    [propget, id(2), helpstring("property BuildId")]
        HRESULT BuildId([out, retval] BSTR *pVal);
    [propget, id(3), helpstring("property Name")]
        HRESULT Name([out, retval] BSTR *pVal);
    [propget, id(4), helpstring("property UserText")]
        HRESULT UserText(long Index, [out, retval] BSTR *pVal);
    [propput, id(4), helpstring("property UserText")]
        HRESULT UserText(long Index, [in] BSTR newVal);
    [propget, id(5), helpstring("property Session")]
        HRESULT Session(long Index, [out, retval] IDispatch* *pVal);
    [propget, id(6), helpstring("property Copyright")]
        HRESULT Copyright([out, retval] BSTR *pVal);
    [propget, id(7), helpstring("property ActiveSessionIndex")]
        HRESULT ActiveSessionIndex([out, retval] long *pVal);
    [propput, id(7), helpstring("property ActiveSessionIndex")]
        HRESULT ActiveSessionIndex([in] long newVal);
    [propget, id(8), helpstring("property ActiveSession")]
        HRESULT ActiveSession([out, retval] IDispatch* *pVal);
    [id(9), helpstring("method SendMenuCommand")]
        HRESULT SendMenuCommand(long CommandId);
};
```



IEXTERNALDEVICE:IDISPATCH

```
interface IExternalDevice : IDispatch
{
[propget, id(1), helpstring("property BatteryStatus")]
    HRESULT BatteryStatus([out, retval] BSTR *pVal);
[propget, id(2), helpstring("property DeviceId")] HRESULT DeviceId([out, retval] BSTR *pVal);
[propget, id(3), helpstring("property IPAddress")] HRESULT IPAddress([out, retval] BSTR *pVal);
[propget, id(4), helpstring("property MACAddress")] HRESULT MACAddress([out, retval] BSTR *pVal);
[propget, id(5), helpstring("property PlatformId")] HRESULT PlatformId([out, retval] BSTR *pVal);
[propget, id(6), helpstring("property PresetId")] HRESULT PresetId([out, retval] BSTR *pVal);
[propget, id(7), helpstring("property SerialNumber")] HRESULT SerialNumber([out, retval] BSTR *pVal);
[propget, id(8), helpstring("property battery")] HRESULT battery([out, retval] IDispatch* *pVal);
[propget, id(9), helpstring("property rf")] HRESULT rf([out, retval] IDispatch* *pVal);
[propget, id(10), helpstring("property UUID")] HRESULT UUID([out, retval] BSTR *pVal);
[propget, id(11), helpstring("property NetworkAdapterName")]
    HRESULT NetworkAdapterName([out, retval] BSTR *pVal);
[propget, id(12), helpstring("property Scanner")] HRESULT Scanner([out, retval] IDispatch* *pVal);
[id(13), helpstring("method Test1")] HRESULT Test1(IDispatch *pDisp);
[id(14), helpstring("method Vibrate")] HRESULT Vibrate(LONG DurationMS);
[id(15), helpstring("method SetTaskBarEnabled")] HRESULT SetTaskBarEnabled(LONG bEnable);
[id(16), helpstring("method SetTaskBarVisible")] HRESULT SetTaskBarVisible(LONG bEnable);
[id(17), helpstring("method SetStartButtonVisible")] HRESULT SetStartButtonVisible(LONG bEnable);
[id(18), helpstring("method SetSIPButtonVisible")] HRESULT SetSIPButtonVisible(LONG bEnable);
[propget, id(19), helpstring("property APIVersion")] HRESULT APIVersion([out, retval] BSTR *pVal);
};
```



IEXTERNALSESSION:IDISPATCH

```
interface IExternalSession : IDispatch
{
[propget, id(1), helpstring("property Host")] HRESULT Host([out, retval] BSTR *pVal);
[propget, id(2), helpstring("property Port")] HRESULT Port([out, retval] long *pVal);
[propget, id(3), helpstring("property Type")] HRESULT Type([out, retval] BSTR *pVal);
[propget, id(4), helpstring("property Path")] HRESULT Path([out, retval] BSTR *pVal);
[propget, id(5), helpstring("property HomePage")] HRESULT HomePage([out, retval] BSTR *pVal);
[propput, id(5), helpstring("property HomePage")] HRESULT HomePage([in] BSTR newVal);
[propput, id(6), helpstring("property Printer")] HRESULT Printer([out, retval] IDispatch* *pval);
};
```

IEXTERNAL:PRINTSTRING

```
interface IExternalPrintString : IDispatch
{
[id(1), helpstring("method PrintString")] HRESULT PrintString(BSTR Data);
};
```

IEXTERNALBATTERY:IDISPATCH

```
interface IExternalBattery : IDispatch
{
[propget, id(1), helpstring("property StatusText")] HRESULT StatusText([out, retval] BSTR *pVal);
[propget, id(2), helpstring("property Status")] HRESULT Status([out, retval] long *pVal);
[propget, id(3), helpstring("property Level")] HRESULT Level([out, retval] long *pVal);
};
```

IEXTERNALRF:IDISPATCH

```
interface IExternalRF : IDispatch
{
[propget, id(1), helpstring("property Strength")] HRESULT Strength([out, retval] long *pVal);
[propget, id(2), helpstring("property Status")] HRESULT Status([out, retval] long *pVal);
};
```

IEXTERNALSCANNER:IDISPATCH

```
interface IExternalScanner : IDispatch
{
[propget, id(1), helpstring("property LXScanner")] HRESULT LXScanner([out, retval] IDispatch* *pVal);
};
```



IEXTERNALDEVICELXESCANNER:IDISPATCH

```
interface IExternalLXEScanner : IDispatch
{
[id(1), helpstring("method LXEScanner")]
HRESULT LXEScanner(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(2), helpstring("method LXEScannerIntrRFID")]
HRESULT LXEScannerIntrRFID(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(3), helpstring("method LXEScannerDisable")]
HRESULT LXEScannerDisable(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(4), helpstring("method LXEScannerEnable")]
HRESULT LXEScannerEnable(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(5), helpstring("method LXEScannerInputDisable")]
HRESULT LXEScannerInputDisable(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(6), helpstring("method LXEScannerInputEnable")]
HRESULT LXEScannerInputEnable(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(7), helpstring("method LXEScannerReset")]
HRESULT LXEScannerReset(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(8), helpstring("method LXEScannerForcePower")]
HRESULT LXEScannerForcePower(LONG port, LONG val, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(9), helpstring("method LXEScannerForceScan")]
HRESULT LXEScannerForceScan(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(10), helpstring("method LXEScannerStart")]
HRESULT LXEScannerStart(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(11), helpstring("method LXEScannerStop")]
HRESULT LXEScannerStop(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(12), helpstring("method LXEScannerGetStatus")]
HRESULT LXEScannerGetStatus(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(13), helpstring("method LXEScannerGetData")]
HRESULT LXEScannerGetData(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(14), helpstring("method LXEScannerPutData")]
HRESULT LXEScannerPutData(BSTR Data, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(15), helpstring("method LXEScannerPowerOn")]
HRESULT LXEScannerPowerOn(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(16), helpstring("method LXEScannerPowerOff")]
HRESULT LXEScannerPowerOff(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(17), helpstring("method LXEScannerKeysOff")]
HRESULT LXEScannerKeysOff(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(18), helpstring("method LXEScannerKeysOn")]
HRESULT LXEScannerKeysOn(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(19), helpstring("method LXEScannerKeyStatus")]
HRESULT LXEScannerKeyStatus(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(20), helpstring("method LXEScannerAttachPort")]
HRESULT LXEScannerAttachPort(LONG port, LONG val, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(21), helpstring("method LXEScannerSetKey")]
HRESULT LXEScannerSetKey(LONG port, LONG val, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(22), helpstring("method LXEScannerSetSerial")]
HRESULT LXEScannerSetSerial(LONG port, LONG baud, LONG data, LONG parity, LONG stop, IDispatch *pDisp,
[out, retval]LONG *pnResult);
[id(23), helpstring("method LXEScannerSetPower")]
HRESULT LXEScannerSetPower(LONG port, LONG val, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(24), helpstring("method LXEScannerStripLead")]
HRESULT LXEScannerStripLead(LONG bEnable, LONG Cnt, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(25), helpstring("method LXEScannerStripTrail")]
HRESULT LXEScannerStripTrail(LONG bEnable, LONG Cnt, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(26), helpstring("method LXEScannerCtrlCodeOff")]
HRESULT LXEScannerCtrlCodeOff(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(27), helpstring("method LXEScannerCtrlCodeOn")]
HRESULT LXEScannerCtrlCodeOn(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(28), helpstring("method LXEScannerPrefix")]
HRESULT LXEScannerPrefix(LONG bEnable, BSTR PrefixText, IDispatch *pDisp, [out, retval]LONG
*pnResult);
[id(29), helpstring("method LXEScannerSuffix")]
HRESULT LXEScannerSuffix(LONG bEnable, BSTR PrefixText, IDispatch *pDisp, [out, retval]LONG
*pnResult);
[id(30), helpstring("method LXEScannerOption")]
HRESULT LXEScannerOption(LONG nOption, LONG nCode, LONG nCodeEx, LONG bEnable, IDispatch *pDisp, [out,
retval]LONG *pnResult);
[id(31), helpstring("method LXEScannerIsOptionEnabled")]
HRESULT LXEScannerIsOptionEnabled(LONG nOption, LONG nCode, LONG nCodeEx, IDispatch *pDisp, [out,
retval]LONG *pnResult);
```



```
[id(32), helpstring("method LXScannerGetParam")]
HRESULT LXScannerGetParam(LONG param, LONG flags, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(33), helpstring("method LXScannerSetParam")]
HRESULT LXScannerSetParam(LONG param, LONG val, LONG flags, IDispatch *pDisp, [out, retval]LONG
*pnResult);
[id(34), helpstring("method LXScannerSymStripLead")]
HRESULT LXScannerSymStripLead(BSTR symb, LONG bEnable, LONG Cnt, IDispatch *pDisp, [out, retval]LONG
*pnResult);
[id(35), helpstring("method LXScannerSymStripTrail")]
HRESULT LXScannerSymStripTrail(BSTR symb, LONG bEnable, LONG Cnt, IDispatch *pDisp, [out, retval]LONG
*pnResult);
[id(36), helpstring("method LXScannerSymPrefix")]
HRESULT LXScannerSymPrefix(BSTR symb, LONG bEnable, BSTR Prefix, IDispatch *pDisp, [out, retval]LONG
*pnResult);
[id(37), helpstring("method LXScannerSymSuffix")]
HRESULT LXScannerSymSuffix(BSTR symb, LONG bEnable, BSTR Suffix, IDispatch *pDisp, [out, retval]LONG
*pnResult);
[id(38), helpstring("method LXScannerSymEnable")]
HRESULT LXScannerSymEnable(BSTR symb, LONG bEnable, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(39), helpstring("method LXScannerSymIsEnabled")]
HRESULT LXScannerSymIsEnabled(BSTR symb, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(40), helpstring("method LXScannerSymGetConfig")]
HRESULT LXScannerSymGetConfig(BSTR symb, IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(41), helpstring("method LXScannerSymSetConfig")]
HRESULT LXScannerSymSetConfig(IDispatch *pDisp, [out, retval]LONG *pnResult);
[id(42), helpstring("method LXScannerSymRemove")]
HRESULT LXScannerSymRemove(BSTR symb, IDispatch *pDisp, [out, retval]LONG *pnResult);
};
```

IEXTERNALINTERFACE:IDISPATCH

```
interface IExternalInterface : IDispatch
{
[id(1), helpstring("method CreateParameterBlock")] HRESULT CreateParameterBlock([out,
retval]IDispatch* *params);
};
```

IEXTERNALJSPARAMETERS:IDISPATCH

```
interface IExternalJSParameters : IDispatch
{
[propget, id(1), helpstring("property Int1")] HRESULT Int1([out, retval] long *pVal);
[propget, id(1), helpstring("property Int1")] HRESULT Int1([in] long newVal);
[propget, id(2), helpstring("property Int2")] HRESULT Int2([out, retval] long *pVal);
[propget, id(2), helpstring("property Int2")] HRESULT Int2([in] long newVal);
[propget, id(3), helpstring("property String1")] HRESULT String1([out, retval] BSTR *pVal);
[propget, id(3), helpstring("property String1")] HRESULT String1([in] BSTR newVal);
[propget, id(4), helpstring("property LastError")] HRESULT LastError([out, retval] long *pVal);
[propget, id(4), helpstring("property LastError")] HRESULT LastError([in] long newVal);
[propget, id(5), helpstring("property String2")] HRESULT String2([out, retval] BSTR *pVal);
[propget, id(5), helpstring("property String2")] HRESULT String2([in] BSTR newVal);
[propget, id(6), helpstring("property Int3")] HRESULT Int3([out, retval] long *pVal);
[propget, id(6), helpstring("property Int3")] HRESULT Int3([in] long newVal);
[propget, id(7), helpstring("property Int4")] HRESULT Int4([out, retval] long *pVal);
[propget, id(7), helpstring("property Int4")] HRESULT Int4([in] long newVal);
[propget, id(8), helpstring("property Int5")] HRESULT Int5([out, retval] long *pVal);
[propget, id(8), helpstring("property Int5")] HRESULT Int5([in] long newVal);
[propget, id(9), helpstring("property Int6")] HRESULT Int6([out, retval] long *pVal);
[propget, id(9), helpstring("property Int6")] HRESULT Int6([in] long newVal);
[propget, id(10), helpstring("property Int7")] HRESULT Int7([out, retval] long *pVal);
[propget, id(10), helpstring("property Int7")] HRESULT Int7([in] long newVal);
[propget, id(11), helpstring("property Int8")] HRESULT Int8([out, retval] long *pVal);
[propget, id(11), helpstring("property Int8")] HRESULT Int8([in] long newVal);
[propget, id(12), helpstring("property Int9")] HRESULT Int9([out, retval] long *pVal);
[propget, id(12), helpstring("property Int9")] HRESULT Int9([in] long newVal);
[propget, id(13), helpstring("property Int10")] HRESULT Int10([out, retval] long *pVal);
[propget, id(13), helpstring("property Int10")] HRESULT Int10([in] long newVal);
[propget, id(14), helpstring("property String3")] HRESULT String3([out, retval] BSTR *pVal);
[propget, id(14), helpstring("property String3")] HRESULT String3([in] BSTR newVal);
};
```



GSW Programming Structures/Values/Constants

GSW = {};		
GSW.GSWBrowse = {};		
GSW.GSWBrowse.Scanner = {};		
GSW.GSWBrowse.Scanner.LXE = {};		
GSW.GSWBrowse.IE = {};		
GSW.GSWBrowse.Menus = {};		

GSW.GSWBrowse.IE		
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_DIALOG		0x00000001
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_DISABLE_HELP_MENU		0x00000002
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_NO3DBORDER		0x00000004
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_SCROLL_NO		0x00000008
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_DISABLE_SCRIPT_INACTIVE		0x00000010
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_OPENNEWWIN		0x00000020
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_DISABLE_OFFSCREEN		0x00000040
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_FLAT_SCROLLBAR		0x00000080
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_DIV_BLOCKDEFAULT		0x00000100
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_ACTIVATE_CLIENTHIT_ONLY		0x00000200
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_OVERRIDEBEHAVIORFACTORY		0x00000400
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_CODEPAGELINKEDFONTS		0x00000800
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_URL_ENCODING_DISABLE_UTF8		0x00001000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_URL_ENCODING_ENABLE_UTF8		0x00002000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_ENABLE_FORMS_AUTOCOMPLETE		0x00004000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_ENABLE_INPLACE_NAVIGATION		0x00010000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_IME_ENABLE_RECONVERSION		0x00020000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_THEME		0x00040000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_NOTHEME		0x00080000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_NOPICS		0x00100000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_NO3DOUTERBORDER		0x00200000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_DISABLE_EDIT_NS_FIXUP		0x00400000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_LOCAL_MACHINE_ACCESS_CHECK		0x00800000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_DISABLE_UNTRUSTEDPROTOCOL		0x01000000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_HOST_NAVIGATES		0x02000000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_ENABLE_REDIRECT_NOTIFICATION		0x04000000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_USE_WINDOWLESS_SELECTCONTROL		0x08000000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_USE_WINDOWED_SELECTCONTROL		0x10000000
GSW.GSWBrowse.IE.DOCHOSTUIFLAG_ENABLE_ACTIVEX_INACTIVATE_MODE		0x20000000



GSW.GSWBrowse.IE - continued	
GSW.GSWBrowse.IE.DLCTL_DLIMAGES	0x00000010
GSW.GSWBrowse.IE.DLCTL_VIDEOS	0x00000020
GSW.GSWBrowse.IE.DLCTL_BGSOUNDS	0x00000040
GSW.GSWBrowse.IE.DLCTL_NO_SCRIPTS	0x00000080
GSW.GSWBrowse.IE.DLCTL_NO_JAVA	0x00000100
GSW.GSWBrowse.IE.DLCTL_NO_RUNACTIVEXCTLS	0x00000200
GSW.GSWBrowse.IE.DLCTL_NO_DLACTIVEXCTLS	0x00000400
GSW.GSWBrowse.IE.DLCTL_DOWNLOADONLY	0x00000800
GSW.GSWBrowse.IE.DLCTL_NO_FRAMEDOWNLOAD	0x00001000
GSW.GSWBrowse.IE.DLCTL_RESYNCHRONIZE	0x00002000
GSW.GSWBrowse.IE.DLCTL_PRAGMA_NO_CACHE	0x00004000
GSW.GSWBrowse.IE.DLCTL_NO_BEHAVIORS	0x00008000
GSW.GSWBrowse.IE.DLCTL_NO_METACHARSET	0x00010000
GSW.GSWBrowse.IE.DLCTL_URL_ENCODING_DISABLE_UTF8	0x00020000
GSW.GSWBrowse.IE.DLCTL_URL_ENCODING_ENABLE_UTF8	0x00040000
GSW.GSWBrowse.IE.DLCTL_NOFRAMES	0x00080000
GSW.GSWBrowse.IE.DLCTL_FORCEOFFLINE	0x10000000
GSW.GSWBrowse.IE.DLCTL_NO_CLIENTPULL	0x20000000
GSW.GSWBrowse.IE.DLCTL_SILENT	0x40000000
GSW.GSWBrowse.IE.DLCTL_OFFLINEIFNOTCONNECTED	0x80000000
GSW.GSWBrowse.IE.DLCTL_OFFLINE	GSW.GSWBrowse.IE.DLCTL_OFFLINEIFNOTCONNECTED

GSW.GSWBrowse.IE.CONTEXT_MENU_DEFAULT	1 << 0
GSW.GSWBrowse.IE.CONTEXT_MENU_IMAGE	1 << 1
GSW.GSWBrowse.IE.CONTEXT_MENU_CONTROL	1 << 2
GSW.GSWBrowse.IE.CONTEXT_MENU_TABLE	1 << 3
GSW.GSWBrowse.IE.CONTEXT_MENU_TEXTSELECT	1 << 4
GSW.GSWBrowse.IE.CONTEXT_MENU_ANCHOR	1 << 5
GSW.GSWBrowse.IE.CONTEXT_MENU_UNKNOWN	1 << 6



LXE Device Constants

Wedge Attachment Port		Value
GSW.GSWBrowse.Scanner.LXE.WEDGEPORT1		1
GSW.GSWBrowse.Scanner.LXE.WEDGEPORT2		2
GSW.GSWBrowse.Scanner.LXE.WEDGEPORT3		3 * Mx7x Only

Scan Keys		
GSW.GSWBrowse.Scanner.LXE.SCANKEYLEFT		1
GSW.GSWBrowse.Scanner.LXE.SCANKEYRIGHT		2
GSW.GSWBrowse.Scanner.LXE.SCANKEYTRIGGER		3

Scanner Port		
GSW.GSWBrowse.Scanner.LXE.SCANPORT_DISABLED		0
GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM1		1
GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM2		2
GSW.GSWBrowse.Scanner.LXE.SCANPORT_COM3		3
GSW.GSWBrowse.Scanner.LXE.SCANPORT_INTERNAL		4
GSW.GSWBrowse.Scanner.LXE.SCANPORT_RFID		5
GSW.GSWBrowse.Scanner.LXE.SCANPORT_BLUETOOTH		6

Validity Parsing		
GSW.GSWBrowse.Scanner.LXE.SCANPORT_MIN		0
GSW.GSWBrowse.Scanner.LXE.SCANPORT_MAX		6

Key Actions		
GSW.GSWBrowse.Scanner.LXE.SCANKEY_DISABLED		0
GSW.GSWBrowse.Scanner.LXE.SCANKEY_SCAN		1
GSW.GSWBrowse.Scanner.LXE.SCANKEY_ENTER		2
GSW.GSWBrowse.Scanner.LXE.SCANKEY_TAB		3
GSW.GSWBrowse.Scanner.LXE.SCANKEY_VIRTKEY		4
GSW.GSWBrowse.Scanner.LXE.SCANKEY_FLDEXIT		5
GSW.GSWBrowse.Scanner.LXE.SCANKEY_RFID		6

Limits for Validity Parsing		
GSW.GSWBrowse.Scanner.LXE.SCANKEY_MIN		0
GSW.GSWBrowse.Scanner.LXE.SCANKEY_MAX		6

Values Returned from LXEScannerGetStatus		
GSW.GSWBrowse.Scanner.LXE.SCAN_COMPLETE		0
GSW.GSWBrowse.Scanner.LXE.SCAN_SCANNING		1
GSW.GSWBrowse.Scanner.LXE.SCAN_NOSCAN		2
GSW.GSWBrowse.Scanner.LXE.SCAN_ERROR		3
GSW.GSWBrowse.Scanner.LXE.SCAN_BUSY		4



Bitmap Values for Scan Code Types

Symbol or Generic (nCode)		
GSW.GSWBrowse.Scanner.LXE.CODE_BOOKLAND		(1 << 0)
GSW.GSWBrowse.Scanner.LXE.CODE_CODABAR		(1 << 1)
GSW.GSWBrowse.Scanner.LXE.CODE_CODE11		(1 << 2)
GSW.GSWBrowse.Scanner.LXE.CODE_CODE39		(1 << 3)
GSW.GSWBrowse.Scanner.LXE.CODE_CODE93		(1 << 4)
GSW.GSWBrowse.Scanner.LXE.CODE_CODE128		(1 << 5)
GSW.GSWBrowse.Scanner.LXE.CODE_COUPON		(1 << 6)
GSW.GSWBrowse.Scanner.LXE.CODE_DISC20F5		(1 << 7)
GSW.GSWBrowse.Scanner.LXE.CODE_EAN8		(1 << 8)
GSW.GSWBrowse.Scanner.LXE.CODE_EAN13		(1 << 9)
GSW.GSWBrowse.Scanner.LXE.CODE_EAN128		(1 << 10)
GSW.GSWBrowse.Scanner.LXE.CODE_FULLASCII		(1 << 11)
GSW.GSWBrowse.Scanner.LXE.CODE_IATA20F5		(1 << 12)
GSW.GSWBrowse.Scanner.LXE.CODE_INTER20F5		(1 << 13)
GSW.GSWBrowse.Scanner.LXE.CODE_ISBT128		(1 << 14)
GSW.GSWBrowse.Scanner.LXE.CODE_MSI		(1 << 15)
GSW.GSWBrowse.Scanner.LXE.CODE_UPDF417		(1 << 16)
GSW.GSWBrowse.Scanner.LXE.CODE_PDF417		(1 << 17)
GSW.GSWBrowse.Scanner.LXE.CODE_PLESSEY		(1 << 18)
GSW.GSWBrowse.Scanner.LXE.CODE_RSS14		(1 << 19)
GSW.GSWBrowse.Scanner.LXE.CODE_RSSEXP		(1 << 20)
GSW.GSWBrowse.Scanner.LXE.CODE_RSSLIM		(1 << 21)
GSW.GSWBrowse.Scanner.LXE.CODE_TRIOP39		(1 << 22)
GSW.GSWBrowse.Scanner.LXE.CODE_UPCA		(1 << 23)
GSW.GSWBrowse.Scanner.LXE.CODE_UPCE		(1 << 24)
GSW.GSWBrowse.Scanner.LXE.CODE_UPCE1		(1 << 25)

Intermec Specific (nCode)		
GSW.GSWBrowse.Scanner.LXE.CODE_CODABLKA		(1 << 26)
GSW.GSWBrowse.Scanner.LXE.CODE_CODABLK		(1 << 27)
GSW.GSWBrowse.Scanner.LXE.CODE_LABELCODE		(1 << 28)
GSW.GSWBrowse.Scanner.LXE.CODE_MATR20F5		(1 << 29)
GSW.GSWBrowse.Scanner.LXE.CODE_STAND20F5		(1 << 30)
GSW.GSWBrowse.Scanner.LXE.CODE_TELEPEN		(1 << 31)

Hand Held Specific (nCode)		
GSW.GSWBrowse.Scanner.LXE.CODE2_AUSPOST		(1 << 0)
GSW.GSWBrowse.Scanner.LXE.CODE2_AZTEC		(1 << 1)
GSW.GSWBrowse.Scanner.LXE.CODE2_BP0		(1 << 2)
GSW.GSWBrowse.Scanner.LXE.CODE2_CANPOST		(1 << 3)
GSW.GSWBrowse.Scanner.LXE.CODE2_CODE49		(1 << 4)
GSW.GSWBrowse.Scanner.LXE.CODE2_COMPOSITE		(1 << 5)
GSW.GSWBrowse.Scanner.LXE.CODE2_DATAMATRIX		(1 << 6)
GSW.GSWBrowse.Scanner.LXE.CODE2_DUTCHPOST		(1 << 7)
GSW.GSWBrowse.Scanner.LXE.CODE2_JAPOST		(1 << 8)
GSW.GSWBrowse.Scanner.LXE.CODE2_MAXICODE		(1 << 9)
GSW.GSWBrowse.Scanner.LXE.CODE2_MESA		(1 << 10)
GSW.GSWBrowse.Scanner.LXE.CODE2_OCR		(1 << 11)
GSW.GSWBrowse.Scanner.LXE.CODE2_POSTNET		(1 << 12)



GSW.GSWBrowse.Scanner.LXE.CODE2_QR	(1 << 13)
GSW.GSWBrowse.Scanner.LXE.CODE2_PLANET	(1 << 14)

Symbologies

ALL Symbologies	
GSW.GSWBrowse.Scanner.LXE.SYMB_ALL	ALL

Symbol and Generic Symbologies	
GSW.GSWBrowse.Scanner.LXE.SYMB_BooklandEAN	BooklandEAN
GSW.GSWBrowse.Scanner.LXE.SYMB_Codabar	Codabar
GSW.GSWBrowse.Scanner.LXE.SYMB_Code11	Code11
GSW.GSWBrowse.Scanner.LXE.SYMB_Code39	Code39
GSW.GSWBrowse.Scanner.LXE.SYMB_Code93	Code93
GSW.GSWBrowse.Scanner.LXE.SYMB_Code128	Code128
GSW.GSWBrowse.Scanner.LXE.SYMB_Coupon	Coupon
GSW.GSWBrowse.Scanner.LXE.SYMB_Discr2of5	Discr2of5
GSW.GSWBrowse.Scanner.LXE.SYMB_Interlv2of5	Interlv2of5
GSW.GSWBrowse.Scanner.LXE.SYMB_MSI_Plessey	MSI/Plessey
GSW.GSWBrowse.Scanner.LXE.SYMB_Trioptic39	Trioptic39
GSW.GSWBrowse.Scanner.LXE.SYMB_UCC_EAN128	UCC/EAN128
GSW.GSWBrowse.Scanner.LXE.SYMB_UPC_EAN	UPC/EAN

Intermec Symbologies	
GSW.GSWBrowse.Scanner.LXE.SYMB_CodeBlkA	CodeBlkA
GSW.GSWBrowse.Scanner.LXE.SYMB_CodaBlkB	CodaBlkB
GSW.GSWBrowse.Scanner.LXE.SYMB_LabelCode	LabelCode
GSW.GSWBrowse.Scanner.LXE.SYMB_Matr2of5	Matr2of5
GSW.GSWBrowse.Scanner.LXE.SYMB_Stand2of5	Stand2of5
GSW.GSWBrowse.Scanner.LXE.SYMB_TelePen	TelePen

Hand Held Products Symbologies	
GSW.GSWBrowse.Scanner.LXE.SYMB_AusPost	AusPost
GSW.GSWBrowse.Scanner.LXE.SYMB_Aztec	Aztec
GSW.GSWBrowse.Scanner.LXE.SYMB_BPO	BPO
GSW.GSWBrowse.Scanner.LXE.SYMB_CanPost	CanPost
GSW.GSWBrowse.Scanner.LXE.SYMB_Code49	Code49
GSW.GSWBrowse.Scanner.LXE.SYMB_Composite	Composite
GSW.GSWBrowse.Scanner.LXE.SYMB_DataMatrix	DataMatrix
GSW.GSWBrowse.Scanner.LXE.SYMB_DutchPost	DutchPost
GSW.GSWBrowse.Scanner.LXE.SYMB_JaPost	JaPost
GSW.GSWBrowse.Scanner.LXE.SYMB_Maxicode	Maxicode
GSW.GSWBrowse.Scanner.LXE.SYMB_Mesa	Mesa
GSW.GSWBrowse.Scanner.LXE.SYMB_OCR	OCR
GSW.GSWBrowse.Scanner.LXE.SYMB_Postnet	Postnet
GSW.GSWBrowse.Scanner.LXE.SYMB_QR	QR
GSW.GSWBrowse.Scanner.LXE.SYMB_Planet	Planet



Parms for LXEScannerOption

GSW.GSWBrowse.Scanner.LXE.SCAN_CODE_ENAB		0
GSW.GSWBrowse.Scanner.LXE.SCAN_AIMID		1

Flag Parms for LXEScanner Get/Set Param

GSW.GSWBrowse.Scanner.LXE.FLAG_SYMBOL		1
GSW.GSWBrowse.Scanner.LXE.FLAG_IMAGER		2



Virtual Key Codes

Symbolic Constant	Hexadecimal Value		
VK_BACK	0x08		
VK_TAB	0x09		
VK_CLEAR	0x0C		
VK_RETURN	0x0D		
VK_SHIFT	0x10		
VK_CONTROL	0x11		
VK_MENU	0x12		
VK_PAUSE	0x13		
VK_CAPITAL	0x14		
VK_ESCAPE	0x1B		
VK_SPACE	0x20		
VK_PRIOR	0x21		
VK_NEXT	0x22		
VK_END	0x23		
VK_HOME	0x24		
VK_LEFT	0x25		
VK_UP	0x26		
VK_RIGHT	0x27		
VK_DOWN	0x28		
VK_SELECT	0x29		
VK_EXECUTE	0x2B		
VK_SNAPSHOT	0x2C		
VK_INSERT	0x2D		
VK_DELETE	0x2E		
VK_HELP	0x2F		
VK_0	0x30		
VK_1	0x31		
VK_2	0x32		
VK_3	0x33		
VK_4	0x34		
VK_5	0x35		
VK_6	0x36		
VK_7	0x37		
VK_8	0x38		
VK_9	0x39		
VK_A	0x41		
VK_B	0x42		
VK_C	0x43		
VK_D	0x44		
VK_E	0x45		
VK_F	0x46		
VK_G	0x47		
VK_H	0x48		
VK_I	0x49		



Symbolic Constant	Hexadecimal Value	Mouse or Keyboard Equivalent	
VK_J	0x4A		
VK_K	0x4B		
VK_L	0x4C		
VK_M	0x4D		
VK_N	0x4E		
VK_O	0x4F		
VK_P	0x50		
VK_Q	0x51		
VK_R	0x52		
VK_S	0x53		
VK_T	0x54		
VK_U	0x55		
VK_V	0x56		
VK_W	0x57		
VK_X	0x58		
VK_Y	0x59		
VK_Z	0x5A		
VK_NUMPAD0	0x60		
VK_NUMPAD1	0x61		
VK_NUMPAD2	0x62		
VK_NUMPAD3	0x63		
VK_NUMPAD4	0x64		
VK_NUMPAD5	0x65		
VK_NUMPAD6	0x66		
VK_NUMPAD7	0x67		
VK_NUMPAD8	0x68		
VK_NUMPAD9	0x69		
VK_MULTIPLY	0x6A		
VK_ADD	0x6B		
VK_SEPARATOR	0x6C		
VK_SUBTRACT	0x6D		
VK_DECIMAL	0x6E		
VK_DIVIDE	0x6F		
VK_F1	0x70		
VK_F2	0x71		
VK_F3	0x72		
VK_F4	0x73		
VK_F5	0x74		
VK_F6	0x75		
VK_F7	0x76		
VK_F8	0x77		
VK_F9	0x78		
VK_F10	0x79		
VK_F11	0x7A		
VK_F12	0x7B		
VK_F13	0x7C		
VK_F14	0x7D		
VK_F15	0x7E		
VK_F16	0x7F		
VK_F17	0x80		



Symbolic Constant	Hexadecimal Value	Mouse or Keyboard Equivalent	
VK_F18	0x81		
VK_F19	0x82		
VK_F20	0x83		
VK_F21	0x84		
VK_F22	0x85		
VK_F23	0x86		
VK_F24	0x87		



Important Files

GSWBROWSE.TXT

NOTE: Line numbers are shown for reference only. They do not exist in the actual file.

```
1. <settings>
2. <License>
3. <LicenseServer>192.168.1.102</LicenseServer>
4. <LicenseServerPage>/gsbrowse/licensing/license.asp</LicenseServerPage>
5. <LicenseFileURL>http://192.168.1.102/gsbrowse/meta/license.xml</LicenseFileURL>
6. </License>
7. <Browser0>
8. <DLDCtrl/><CtxMenuControl/><View><HomePage>http://192.168.1.102/gsbrowse/meta/default.htm</HomePage><Caption>0</Caption></View>
9. </Browser0>
10. <Browser1>
11. <View><HomePage>http://192.168.1.102/gsbrowse/meta/timernav.htm</HomePage><DLDCtrl/><CtxMenuControl/></View>
12. </Browser1>
13. <Browser2>
14. <View><HomePage>http://192.168.1.102/gsbrowse/meta/batt_nav.htm</HomePage><DLDCtrl/><CtxMenuControl/></View>
15. </Browser2>
16. <Browser3>
17. <View><HomePage>http://192.168.1.102/gsbrowse/meta/onkey.htm</HomePage><DLDCtrl/><CtxMenuControl/></View>
18. </Browser3>
19. <Browser4>
20. <View><HomePage>http://192.168.1.102/gsbrowse/meta/external.htm</HomePage><DLDCtrl/><CtxMenuControl/></View>
21. </Browser4>
22. <UI><ActiveBrowser>1</ActiveBrowser></UI></settings>
```



LICENSE.ASP

NOTE: Line numbers are shown for reference only. They do not exist in the actual file.

The example below is a working Post Acceptor ASP example.

```
1. <!-- #include file="WriteLog.asp" -->
2. <%
    WriteToLog Request("MAC") & "-->" & Request("ProductId")
    CloseLog
3. %>
4. <BODY>
5. <p>This is an internal ASP page</p>
6. </BODY>
```



WRITELOG.ASP

NOTE: Line numbers are shown for reference only. They do not exist in the actual file.

The example below is a working example and the only line you will need to change is line #5 (in blue).

```
1. <%
2. Const ForAppending = 8
3.
4. Dim strLogFileName
5. strLogFileName = "c:\inetpub\wwwroot\gsbrowse\meta\license.log"
6.
7. Dim objLogFileFSO
8. set objLogFileFSO = CreateObject("Scripting.FileSystemObject")
9.
10. Dim objLogFileTS
11. If objLogFileFSO.FileExists(strLogFileName) Then
12.     set objLogFileTS = objLogFileFSO.OpenTextFile(strLogFileName, ForAppending)
13. Else
14.     set objLogFileTS = objLogFileFSO.CreateTextFile(strLogFileName)
15. End If
16.
17. Sub WriteToLog (strNewEntry)
18.     Dim strLogEntry
19.
20.     strLogEntry = FormatDateTime(Now) & " - "
21.     strLogEntry = strLogEntry & strNewEntry
22.     objLogFileTS.WriteLine(strLogEntry)
23. End Sub
24.
25. Sub CloseLog()
26.     objLogFileTS.Close
27. End Sub
28. %>
```



LICENSE.XML

NOTE: Line numbers are shown for reference only. They do not exist in the actual file.

```
1. <?xml version="1.0" ?>
2. <Licensing>
3. <License>
4. <ProductId>3CF4AF6F6F0ACDA44E77002A6F00CA8F57D4C2171346</ProductId>
5. <SerialNumber>D25EEAF8B30C83EF089A5CE0F54844433986632C82D8</SerialNumber>
6. </License>
7. <License>
8. <ProductId>3CF4AF6F6F0ACDA44E77002A6F00CAB17F07CB17ABB7</ProductId>
9. <SerialNumber>D25EEAF8B30C83EF089A5CE0F548447D2586A065A9D8</SerialNumber>
10. </License>
11. <License>
12. <ProductId>3CF4AF6F6F0ACDA44E77002A23FFDAC657D4C2171346</ProductId>
13. <SerialNumber>D25EEAF8B30C83EF089A5CE0B9B7540A1286B96BCCD8</SerialNumber>
14. </License>
15. <License>
16. <ProductId>3CF4AF6F6F0ACDA44E77002A70BF0C0C57D4C2171346</ProductId>
17. <SerialNumber>D25EEAF8B30C83EF089A5CE0EAF782C07388632CC0D8</SerialNumber>
18. </License>
19. <License>
20. <ProductId>3CF4AF6F6F0ACDA44E77002A6F00CA8B57D4C2171346</ProductId>
21. <SerialNumber>D25EEAF8B30C83EF089A5CE0F54844477888632CA0D9</SerialNumber>
22. </License>
23. </Licensing>
```



GSWBrowse Subscription

TBD

How to Update The Software

1. Simply download and install the new version of **TBD**

HOW TO RENEW THE GSWBrowse Software

Please use the following procedure when renewing GSWBrowse Subscription.

TBD



System Signature - IMPORTANT PLEASE READ

NOTE: This section only applies to Software Registration

The registration software obtains a system signature that is unique to your system. This signature is an added security measure to inhibit unauthorized personnel from obtaining working copies of the GSWBrowse .

The signature is comprised of hardware and software identifiers existing on your system which make the target system unique. These identifiers are hashed into a Product ID so a Serial Number can be generated from this Product id.

If major hardware components of your system are removed, replaced or modified your **Serial Number** may discontinue to work and you may need a new **Serial Number** to obtain access to the GSWBrowse. Please contact Georgia SoftWorks Technical Support if needed.



Technical Support

In order to keep Technical Support **Free** please help minimize the cost.

- Gather all relevant system and environment information.
- Write your question down. This not only helps us but also helps you articulate the question better.

If the question is not an emergency, please use e-mail at support@georgiasoftworks.com. We try to respond within 24 hours.

Or Call 706.265.1018 EST, M-F 9:00 a.m. to 5:00 p.m. and have your Product ID ready